

activemq-cpp-3.2.5

Generated by Doxygen 1.6.1

Wed Jul 25 23:57:04 2012



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>25</b>
3.1	Data Structures . . . . .	25
<b>4</b>	<b>File Index</b>	<b>63</b>
4.1	File List . . . . .	63
<b>5</b>	<b>Namespace Documentation</b>	<b>85</b>
5.1	activemq Namespace Reference . . . . .	85
5.1.1	Detailed Description . . . . .	85
5.2	activemq::cmsutil Namespace Reference . . . . .	86
5.3	activemq::commands Namespace Reference . . . . .	87
5.4	activemq::core Namespace Reference . . . . .	89
5.5	activemq::core::policies Namespace Reference . . . . .	91
5.6	activemq::exceptions Namespace Reference . . . . .	92
5.7	activemq::io Namespace Reference . . . . .	93
5.8	activemq::library Namespace Reference . . . . .	94
5.9	activemq::state Namespace Reference . . . . .	95
5.10	activemq::threads Namespace Reference . . . . .	96
5.11	activemq::transport Namespace Reference . . . . .	97
5.12	activemq::transport::correlator Namespace Reference . . . . .	98
5.13	activemq::transport::failover Namespace Reference . . . . .	99
5.14	activemq::transport::inactivity Namespace Reference . . . . .	100
5.15	activemq::transport::logging Namespace Reference . . . . .	101
5.16	activemq::transport::mock Namespace Reference . . . . .	102

5.17	activemq::transport::tcp Namespace Reference . . . . .	103
5.18	activemq::util Namespace Reference . . . . .	104
5.19	activemq::wireformat Namespace Reference . . . . .	105
5.20	activemq::wireformat::openwire Namespace Reference . . . . .	106
5.21	activemq::wireformat::openwire::marshal Namespace Reference . . . . .	107
5.22	activemq::wireformat::openwire::marshal::v1 Namespace Reference . . . . .	108
5.23	activemq::wireformat::openwire::marshal::v2 Namespace Reference . . . . .	113
5.24	activemq::wireformat::openwire::marshal::v3 Namespace Reference . . . . .	118
5.25	activemq::wireformat::openwire::marshal::v4 Namespace Reference . . . . .	123
5.26	activemq::wireformat::openwire::marshal::v5 Namespace Reference . . . . .	128
5.27	activemq::wireformat::openwire::marshal::v6 Namespace Reference . . . . .	133
5.28	activemq::wireformat::openwire::utils Namespace Reference . . . . .	138
5.29	activemq::wireformat::stomp Namespace Reference . . . . .	139
5.30	cms Namespace Reference . . . . .	140
5.30.1	Detailed Description . . . . .	142
5.31	decaf Namespace Reference . . . . .	143
5.31.1	Detailed Description . . . . .	143
5.32	decaf::internal Namespace Reference . . . . .	144
5.33	decaf::internal::io Namespace Reference . . . . .	145
5.34	decaf::internal::net Namespace Reference . . . . .	146
5.35	decaf::internal::net::ssl Namespace Reference . . . . .	147
5.36	decaf::internal::net::ssl::openssl Namespace Reference . . . . .	148
5.37	decaf::internal::net::tcp Namespace Reference . . . . .	149
5.38	decaf::internal::nio Namespace Reference . . . . .	150
5.39	decaf::internal::security Namespace Reference . . . . .	151
5.40	decaf::internal::util Namespace Reference . . . . .	152
5.41	decaf::internal::util::concurrent Namespace Reference . . . . .	153
5.42	decaf::io Namespace Reference . . . . .	154
5.43	decaf::lang Namespace Reference . . . . .	156
5.43.1	Function Documentation . . . . .	158
5.43.1.1	operator!= . . . . .	158
5.43.1.2	operator!= . . . . .	158
5.43.1.3	operator!= . . . . .	158
5.43.1.4	operator!= . . . . .	158
5.43.1.5	operator== . . . . .	158
5.43.1.6	operator== . . . . .	158



5.43.1.7	operator==	158
5.43.1.8	operator==	158
5.44	decaf::lang::exceptions Namespace Reference	159
5.45	decaf::net Namespace Reference	160
5.46	decaf::net::ssl Namespace Reference	162
5.47	decaf::nio Namespace Reference	163
5.48	decaf::security Namespace Reference	164
5.49	decaf::security::auth Namespace Reference	165
5.50	decaf::security::auth::x500 Namespace Reference	166
5.51	decaf::security::cert Namespace Reference	167
5.52	decaf::util Namespace Reference	168
5.53	decaf::util::comparators Namespace Reference	170
5.54	decaf::util::concurrent Namespace Reference	171
5.55	decaf::util::concurrent::atomic Namespace Reference	173
5.56	decaf::util::concurrent::locks Namespace Reference	174
5.57	decaf::util::logging Namespace Reference	175
5.57.1	Enumeration Type Documentation	176
5.57.1.1	Levels	176
5.58	decaf::util::zip Namespace Reference	177
5.59	std Namespace Reference	178
<b>6</b>	<b>Data Structure Documentation</b>	<b>179</b>
6.1	decaf::util::AbstractCollection< E > Class Template Reference	179
6.1.1	Detailed Description	181
6.1.2	Constructor & Destructor Documentation	182
6.1.2.1	AbstractCollection	182
6.1.2.2	~AbstractCollection	182
6.1.3	Member Function Documentation	182
6.1.3.1	add	182
6.1.3.2	addAll	183
6.1.3.3	clear	183
6.1.3.4	contains	184
6.1.3.5	containsAll	185
6.1.3.6	copy	185
6.1.3.7	equals	185
6.1.3.8	isEmpty	186
6.1.3.9	lock	186

6.1.3.10	notify . . . . .	186
6.1.3.11	notifyAll . . . . .	187
6.1.3.12	operator= . . . . .	187
6.1.3.13	remove . . . . .	187
6.1.3.14	removeAll . . . . .	188
6.1.3.15	retainAll . . . . .	189
6.1.3.16	toArray . . . . .	189
6.1.3.17	tryLock . . . . .	189
6.1.3.18	unlock . . . . .	190
6.1.3.19	wait . . . . .	190
6.1.3.20	wait . . . . .	191
6.1.3.21	wait . . . . .	191
6.1.4	Field Documentation . . . . .	191
6.1.4.1	mutex . . . . .	191
6.2	decaf::util::AbstractList< E > Class Template Reference . . . . .	192
6.2.1	Detailed Description . . . . .	192
6.2.2	Constructor & Destructor Documentation . . . . .	192
6.2.2.1	~AbstractList . . . . .	192
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference . . . . .	193
6.3.1	Detailed Description . . . . .	193
6.3.2	Constructor & Destructor Documentation . . . . .	193
6.3.2.1	~AbstractMap . . . . .	193
6.4	decaf::util::AbstractQueue< E > Class Template Reference . . . . .	194
6.4.1	Detailed Description . . . . .	194
6.4.2	Constructor & Destructor Documentation . . . . .	195
6.4.2.1	AbstractQueue . . . . .	195
6.4.2.2	~AbstractQueue . . . . .	195
6.4.3	Member Function Documentation . . . . .	195
6.4.3.1	add . . . . .	195
6.4.3.2	addAll . . . . .	195
6.4.3.3	clear . . . . .	196
6.4.3.4	element . . . . .	196
6.4.3.5	remove . . . . .	196
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference . . . . .	198
6.5.1	Detailed Description . . . . .	198
6.5.2	Constructor & Destructor Documentation . . . . .	198

6.5.2.1	~AbstractSequentialList . . . . .	198
6.6	decaf::util::AbstractSet< E > Class Template Reference . . . . .	199
6.6.1	Detailed Description . . . . .	199
6.6.2	Constructor & Destructor Documentation . . . . .	199
6.6.2.1	~AbstractSet . . . . .	199
6.6.3	Member Function Documentation . . . . .	199
6.6.3.1	removeAll . . . . .	199
6.7	activemq::transport::AbstractTransportFactory Class Reference . . . . .	201
6.7.1	Detailed Description . . . . .	201
6.7.2	Constructor & Destructor Documentation . . . . .	201
6.7.2.1	~AbstractTransportFactory . . . . .	201
6.7.3	Member Function Documentation . . . . .	201
6.7.3.1	createWireFormat . . . . .	201
6.8	activemq::core::ActiveMQAckHandler Class Reference . . . . .	203
6.8.1	Detailed Description . . . . .	203
6.8.2	Constructor & Destructor Documentation . . . . .	203
6.8.2.1	~ActiveMQAckHandler . . . . .	203
6.8.3	Member Function Documentation . . . . .	203
6.8.3.1	acknowledgeMessage . . . . .	203
6.9	activemq::commands::ActiveMQBlobMessage Class Reference . . . . .	204
6.9.1	Constructor & Destructor Documentation . . . . .	205
6.9.1.1	ActiveMQBlobMessage . . . . .	205
6.9.1.2	~ActiveMQBlobMessage . . . . .	205
6.9.2	Member Function Documentation . . . . .	205
6.9.2.1	clone . . . . .	205
6.9.2.2	cloneDataStructure . . . . .	205
6.9.2.3	copyDataStructure . . . . .	205
6.9.2.4	equals . . . . .	206
6.9.2.5	getDataStructureType . . . . .	206
6.9.2.6	getMimeType . . . . .	206
6.9.2.7	getName . . . . .	206
6.9.2.8	getRemoteBlobUrl . . . . .	207
6.9.2.9	isDeletedByBroker . . . . .	207
6.9.2.10	setDeletedByBroker . . . . .	207
6.9.2.11	setMimeType . . . . .	207
6.9.2.12	setName . . . . .	207

6.9.2.13	setRemoteBlobUrl . . . . .	207
6.9.2.14	toString . . . . .	208
6.9.3	Field Documentation . . . . .	208
6.9.3.1	BINARY_MIME_TYPE . . . . .	208
6.9.3.2	ID_ACTIVEMQBLOBMESSAGE . . . . .	208
6.10	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	209
6.10.1	Detailed Description . . . . .	209
6.10.2	Constructor & Destructor Documentation . . . . .	210
6.10.2.1	ActiveMQBlobMessageMarshaller . . . . .	210
6.10.2.2	~ActiveMQBlobMessageMarshaller . . . . .	210
6.10.3	Member Function Documentation . . . . .	210
6.10.3.1	createObject . . . . .	210
6.10.3.2	getDataStructureType . . . . .	210
6.10.3.3	looseMarshal . . . . .	210
6.10.3.4	looseUnmarshal . . . . .	211
6.10.3.5	tightMarshal1 . . . . .	211
6.10.3.6	tightMarshal2 . . . . .	211
6.10.3.7	tightUnmarshal . . . . .	212
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	213
6.11.1	Detailed Description . . . . .	213
6.11.2	Constructor & Destructor Documentation . . . . .	214
6.11.2.1	ActiveMQBlobMessageMarshaller . . . . .	214
6.11.2.2	~ActiveMQBlobMessageMarshaller . . . . .	214
6.11.3	Member Function Documentation . . . . .	214
6.11.3.1	createObject . . . . .	214
6.11.3.2	getDataStructureType . . . . .	214
6.11.3.3	looseMarshal . . . . .	214
6.11.3.4	looseUnmarshal . . . . .	215
6.11.3.5	tightMarshal1 . . . . .	215
6.11.3.6	tightMarshal2 . . . . .	215
6.11.3.7	tightUnmarshal . . . . .	216
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	217
6.12.1	Detailed Description . . . . .	217
6.12.2	Constructor & Destructor Documentation . . . . .	218

6.12.2.1	ActiveMQBlobMessageMarshaller . . . . .	218
6.12.2.2	~ActiveMQBlobMessageMarshaller . . . . .	218
6.12.3	Member Function Documentation . . . . .	218
6.12.3.1	createObject . . . . .	218
6.12.3.2	getDataStructureType . . . . .	218
6.12.3.3	looseMarshal . . . . .	218
6.12.3.4	looseUnmarshal . . . . .	219
6.12.3.5	tightMarshal1 . . . . .	219
6.12.3.6	tightMarshal2 . . . . .	219
6.12.3.7	tightUnmarshal . . . . .	220
6.13	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	221
6.13.1	Detailed Description . . . . .	221
6.13.2	Constructor & Destructor Documentation . . . . .	222
6.13.2.1	ActiveMQBlobMessageMarshaller . . . . .	222
6.13.2.2	~ActiveMQBlobMessageMarshaller . . . . .	222
6.13.3	Member Function Documentation . . . . .	222
6.13.3.1	createObject . . . . .	222
6.13.3.2	getDataStructureType . . . . .	222
6.13.3.3	looseMarshal . . . . .	222
6.13.3.4	looseUnmarshal . . . . .	223
6.13.3.5	tightMarshal1 . . . . .	223
6.13.3.6	tightMarshal2 . . . . .	223
6.13.3.7	tightUnmarshal . . . . .	224
6.14	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	225
6.14.1	Detailed Description . . . . .	225
6.14.2	Constructor & Destructor Documentation . . . . .	226
6.14.2.1	ActiveMQBlobMessageMarshaller . . . . .	226
6.14.2.2	~ActiveMQBlobMessageMarshaller . . . . .	226
6.14.3	Member Function Documentation . . . . .	226
6.14.3.1	createObject . . . . .	226
6.14.3.2	getDataStructureType . . . . .	226
6.14.3.3	looseMarshal . . . . .	226
6.14.3.4	looseUnmarshal . . . . .	227
6.14.3.5	tightMarshal1 . . . . .	227
6.14.3.6	tightMarshal2 . . . . .	227

6.14.3.7	tightUnmarshal . . . . .	228
6.15	activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	229
6.15.1	Detailed Description . . . . .	229
6.15.2	Constructor & Destructor Documentation . . . . .	230
6.15.2.1	ActiveMQBlobMessageMarshaller . . . . .	230
6.15.2.2	~ActiveMQBlobMessageMarshaller . . . . .	230
6.15.3	Member Function Documentation . . . . .	230
6.15.3.1	createObject . . . . .	230
6.15.3.2	getDataStructureType . . . . .	230
6.15.3.3	looseMarshal . . . . .	230
6.15.3.4	looseUnmarshal . . . . .	231
6.15.3.5	tightMarshal1 . . . . .	231
6.15.3.6	tightMarshal2 . . . . .	231
6.15.3.7	tightUnmarshal . . . . .	232
6.16	activemq::commands::ActiveMQBytesMessage Class Reference . . . . .	233
6.16.1	Constructor & Destructor Documentation . . . . .	236
6.16.1.1	ActiveMQBytesMessage . . . . .	236
6.16.1.2	~ActiveMQBytesMessage . . . . .	236
6.16.2	Member Function Documentation . . . . .	236
6.16.2.1	clearBody . . . . .	236
6.16.2.2	clone . . . . .	236
6.16.2.3	cloneDataStructure . . . . .	236
6.16.2.4	copyDataStructure . . . . .	237
6.16.2.5	equals . . . . .	237
6.16.2.6	getBodyBytes . . . . .	237
6.16.2.7	getBodyLength . . . . .	238
6.16.2.8	getDataStructureType . . . . .	238
6.16.2.9	onSend . . . . .	238
6.16.2.10	readBoolean . . . . .	238
6.16.2.11	readByte . . . . .	239
6.16.2.12	readBytes . . . . .	239
6.16.2.13	readBytes . . . . .	240
6.16.2.14	readChar . . . . .	240
6.16.2.15	readDouble . . . . .	240
6.16.2.16	readFloat . . . . .	241
6.16.2.17	readInt . . . . .	241

6.16.2.18 readLong . . . . .	241
6.16.2.19 readShort . . . . .	242
6.16.2.20 readString . . . . .	242
6.16.2.21 readUnsignedShort . . . . .	243
6.16.2.22 readUTF . . . . .	243
6.16.2.23 reset . . . . .	243
6.16.2.24 setBodyBytes . . . . .	244
6.16.2.25 toString . . . . .	244
6.16.2.26 writeBoolean . . . . .	244
6.16.2.27 writeByte . . . . .	244
6.16.2.28 writeBytes . . . . .	245
6.16.2.29 writeBytes . . . . .	245
6.16.2.30 writeChar . . . . .	245
6.16.2.31 writeDouble . . . . .	246
6.16.2.32 writeFloat . . . . .	246
6.16.2.33 writeInt . . . . .	246
6.16.2.34 writeLong . . . . .	247
6.16.2.35 writeShort . . . . .	247
6.16.2.36 writeString . . . . .	247
6.16.2.37 writeUnsignedShort . . . . .	248
6.16.2.38 writeUTF . . . . .	248
6.16.3 Field Documentation . . . . .	248
6.16.3.1 ID_ACTIVEMQBYTESMESSAGE . . . . .	248
6.17 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
Class Reference . . . . .	249
6.17.1 Detailed Description . . . . .	249
6.17.2 Constructor & Destructor Documentation . . . . .	250
6.17.2.1 ActiveMQBytesMessageMarshaller . . . . .	250
6.17.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	250
6.17.3 Member Function Documentation . . . . .	250
6.17.3.1 createObject . . . . .	250
6.17.3.2 getDataStructureType . . . . .	250
6.17.3.3 looseMarshal . . . . .	250
6.17.3.4 looseUnmarshal . . . . .	251
6.17.3.5 tightMarshal1 . . . . .	251
6.17.3.6 tightMarshal2 . . . . .	251
6.17.3.7 tightUnmarshal . . . . .	252

6.18	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	253
6.18.1	Detailed Description . . . . .	253
6.18.2	Constructor & Destructor Documentation . . . . .	254
	6.18.2.1 ActiveMQBytesMessageMarshaller . . . . .	254
	6.18.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	254
6.18.3	Member Function Documentation . . . . .	254
	6.18.3.1 createObject . . . . .	254
	6.18.3.2 getDataStructureType . . . . .	254
	6.18.3.3 looseMarshal . . . . .	254
	6.18.3.4 looseUnmarshal . . . . .	255
	6.18.3.5 tightMarshal1 . . . . .	255
	6.18.3.6 tightMarshal2 . . . . .	255
	6.18.3.7 tightUnmarshal . . . . .	256
6.19	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	257
6.19.1	Detailed Description . . . . .	257
6.19.2	Constructor & Destructor Documentation . . . . .	258
	6.19.2.1 ActiveMQBytesMessageMarshaller . . . . .	258
	6.19.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	258
6.19.3	Member Function Documentation . . . . .	258
	6.19.3.1 createObject . . . . .	258
	6.19.3.2 getDataStructureType . . . . .	258
	6.19.3.3 looseMarshal . . . . .	258
	6.19.3.4 looseUnmarshal . . . . .	259
	6.19.3.5 tightMarshal1 . . . . .	259
	6.19.3.6 tightMarshal2 . . . . .	259
	6.19.3.7 tightUnmarshal . . . . .	260
6.20	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	261
6.20.1	Detailed Description . . . . .	261
6.20.2	Constructor & Destructor Documentation . . . . .	262
	6.20.2.1 ActiveMQBytesMessageMarshaller . . . . .	262
	6.20.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	262
6.20.3	Member Function Documentation . . . . .	262
	6.20.3.1 createObject . . . . .	262
	6.20.3.2 getDataStructureType . . . . .	262



6.20.3.3	looseMarshal . . . . .	262
6.20.3.4	looseUnmarshal . . . . .	263
6.20.3.5	tightMarshal1 . . . . .	263
6.20.3.6	tightMarshal2 . . . . .	263
6.20.3.7	tightUnmarshal . . . . .	264
6.21	activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	265
6.21.1	Detailed Description . . . . .	265
6.21.2	Constructor & Destructor Documentation . . . . .	266
6.21.2.1	ActiveMQBytesMessageMarshaller . . . . .	266
6.21.2.2	~ActiveMQBytesMessageMarshaller . . . . .	266
6.21.3	Member Function Documentation . . . . .	266
6.21.3.1	createObject . . . . .	266
6.21.3.2	getDataStructureType . . . . .	266
6.21.3.3	looseMarshal . . . . .	266
6.21.3.4	looseUnmarshal . . . . .	267
6.21.3.5	tightMarshal1 . . . . .	267
6.21.3.6	tightMarshal2 . . . . .	267
6.21.3.7	tightUnmarshal . . . . .	268
6.22	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	269
6.22.1	Detailed Description . . . . .	269
6.22.2	Constructor & Destructor Documentation . . . . .	270
6.22.2.1	ActiveMQBytesMessageMarshaller . . . . .	270
6.22.2.2	~ActiveMQBytesMessageMarshaller . . . . .	270
6.22.3	Member Function Documentation . . . . .	270
6.22.3.1	createObject . . . . .	270
6.22.3.2	getDataStructureType . . . . .	270
6.22.3.3	looseMarshal . . . . .	270
6.22.3.4	looseUnmarshal . . . . .	271
6.22.3.5	tightMarshal1 . . . . .	271
6.22.3.6	tightMarshal2 . . . . .	271
6.22.3.7	tightUnmarshal . . . . .	272
6.23	activemq::core::ActiveMQConnection Class Reference . . . . .	273
6.23.1	Detailed Description . . . . .	278
6.23.2	Constructor & Destructor Documentation . . . . .	278
6.23.2.1	ActiveMQConnection . . . . .	278

6.23.2.2	~ActiveMQConnection . . . . .	278
6.23.3	Member Function Documentation . . . . .	278
6.23.3.1	addDispatcher . . . . .	278
6.23.3.2	addProducer . . . . .	278
6.23.3.3	addTransportListener . . . . .	278
6.23.3.4	close . . . . .	279
6.23.3.5	createSession . . . . .	279
6.23.3.6	createSession . . . . .	279
6.23.3.7	destroyDestination . . . . .	279
6.23.3.8	destroyDestination . . . . .	280
6.23.3.9	fire . . . . .	280
6.23.3.10	getBrokerURL . . . . .	280
6.23.3.11	getClientID . . . . .	281
6.23.3.12	getCloseTimeout . . . . .	281
6.23.3.13	getConnectionId . . . . .	281
6.23.3.14	getConnectionInfo . . . . .	281
6.23.3.15	getExceptionListener . . . . .	281
6.23.3.16	getMetaData . . . . .	282
6.23.3.17	getNextLocalTransactionId . . . . .	282
6.23.3.18	getNextSessionId . . . . .	282
6.23.3.19	getNextTempDestinationId . . . . .	282
6.23.3.20	getPassword . . . . .	283
6.23.3.21	getPrefetchPolicy . . . . .	283
6.23.3.22	getProducerWindowSize . . . . .	283
6.23.3.23	getRedeliveryPolicy . . . . .	283
6.23.3.24	getSendTimeout . . . . .	283
6.23.3.25	getTransport . . . . .	284
6.23.3.26	getUsername . . . . .	284
6.23.3.27	isAlwaysSyncSend . . . . .	284
6.23.3.28	isClosed . . . . .	284
6.23.3.29	isDispatchAsync . . . . .	284
6.23.3.30	isStarted . . . . .	284
6.23.3.31	isTransportFailed . . . . .	285
6.23.3.32	isUseAsyncSend . . . . .	285
6.23.3.33	isUseCompression . . . . .	285
6.23.3.34	onCommand . . . . .	285

6.23.3.35 oneway . . . . .	285
6.23.3.36 onException . . . . .	286
6.23.3.37 removeDispatcher . . . . .	286
6.23.3.38 removeProducer . . . . .	286
6.23.3.39 removeSession . . . . .	286
6.23.3.40 removeTransportListener . . . . .	286
6.23.3.41 sendPullRequest . . . . .	287
6.23.3.42 setAlwaysSyncSend . . . . .	287
6.23.3.43 setBrokerURL . . . . .	287
6.23.3.44 setClientID . . . . .	287
6.23.3.45 setCloseTimeout . . . . .	288
6.23.3.46 setDefaultClientId . . . . .	288
6.23.3.47 setDispatchAsync . . . . .	288
6.23.3.48 setExceptionListener . . . . .	288
6.23.3.49 setPassword . . . . .	288
6.23.3.50 setPrefetchPolicy . . . . .	289
6.23.3.51 setProducerWindowSize . . . . .	289
6.23.3.52 setRedeliveryPolicy . . . . .	289
6.23.3.53 setSendTimeout . . . . .	289
6.23.3.54 setTransportInterruptProcessingComplete . . . . .	289
6.23.3.55 setUseAsyncSend . . . . .	290
6.23.3.56 setUseCompression . . . . .	290
6.23.3.57 setUsername . . . . .	290
6.23.3.58 start . . . . .	290
6.23.3.59 stop . . . . .	290
6.23.3.60 syncRequest . . . . .	291
6.23.3.61 transportInterrupted . . . . .	291
6.23.3.62 transportResumed . . . . .	291
6.24 activemq::core::ActiveMQConnectionFactory Class Reference . . . . .	292
6.24.1 Constructor & Destructor Documentation . . . . .	294
6.24.1.1 ActiveMQConnectionFactory . . . . .	294
6.24.1.2 ActiveMQConnectionFactory . . . . .	294
6.24.1.3 ~ActiveMQConnectionFactory . . . . .	295
6.24.2 Member Function Documentation . . . . .	295
6.24.2.1 createConnection . . . . .	295
6.24.2.2 createConnection . . . . .	295

6.24.2.3	createConnection . . . . .	296
6.24.2.4	createConnection . . . . .	296
6.24.2.5	getBrokerURL . . . . .	296
6.24.2.6	getClientId . . . . .	297
6.24.2.7	getCloseTimeout . . . . .	297
6.24.2.8	getExceptionListener . . . . .	297
6.24.2.9	getPassword . . . . .	297
6.24.2.10	getPrefetchPolicy . . . . .	297
6.24.2.11	getProducerWindowSize . . . . .	298
6.24.2.12	getRedeliveryPolicy . . . . .	298
6.24.2.13	getSendTimeout . . . . .	298
6.24.2.14	getUsername . . . . .	298
6.24.2.15	isAlwaysSyncSend . . . . .	298
6.24.2.16	isDispatchAsync . . . . .	299
6.24.2.17	isUseAsyncSend . . . . .	299
6.24.2.18	isUseCompression . . . . .	299
6.24.2.19	setAlwaysSyncSend . . . . .	299
6.24.2.20	setBrokerURL . . . . .	299
6.24.2.21	setClientId . . . . .	299
6.24.2.22	setCloseTimeout . . . . .	300
6.24.2.23	setDispatchAsync . . . . .	300
6.24.2.24	setExceptionListener . . . . .	300
6.24.2.25	setPassword . . . . .	300
6.24.2.26	setPrefetchPolicy . . . . .	300
6.24.2.27	setProducerWindowSize . . . . .	301
6.24.2.28	setRedeliveryPolicy . . . . .	301
6.24.2.29	setSendTimeout . . . . .	301
6.24.2.30	setUseAsyncSend . . . . .	301
6.24.2.31	setUseCompression . . . . .	301
6.24.2.32	setUsername . . . . .	302
6.24.3	Field Documentation . . . . .	302
6.24.3.1	DEFAULT_URI . . . . .	302
6.25	activemq::core::ActiveMQConnectionMetaData Class Reference . . . . .	303
6.25.1	Detailed Description . . . . .	303
6.25.2	Constructor & Destructor Documentation . . . . .	304
6.25.2.1	ActiveMQConnectionMetaData . . . . .	304

6.25.2.2	<code>~ActiveMQConnectionMetaData</code> . . . . .	304
6.25.3	Member Function Documentation . . . . .	304
6.25.3.1	<code>getCMSMajorVersion</code> . . . . .	304
6.25.3.2	<code>getCMSMinorVersion</code> . . . . .	304
6.25.3.3	<code>getCMSProviderName</code> . . . . .	304
6.25.3.4	<code>getCMSVersion</code> . . . . .	305
6.25.3.5	<code>getCMSXPropertyNames</code> . . . . .	305
6.25.3.6	<code>getProviderMajorVersion</code> . . . . .	305
6.25.3.7	<code>getProviderMinorVersion</code> . . . . .	306
6.25.3.8	<code>getProviderVersion</code> . . . . .	306
6.26	<code>activemq::core::ActiveMQConstants</code> Class Reference . . . . .	307
6.26.1	Detailed Description . . . . .	308
6.26.2	Member Enumeration Documentation . . . . .	308
6.26.2.1	<code>AckType</code> . . . . .	308
6.26.2.2	<code>DestinationActions</code> . . . . .	308
6.26.2.3	<code>DestinationOption</code> . . . . .	308
6.26.2.4	<code>TransactionState</code> . . . . .	309
6.26.2.5	<code>URIParam</code> . . . . .	309
6.26.3	Member Function Documentation . . . . .	309
6.26.3.1	<code>toDestinationOption</code> . . . . .	309
6.26.3.2	<code>toString</code> . . . . .	309
6.26.3.3	<code>toString</code> . . . . .	309
6.26.3.4	<code>toURIOption</code> . . . . .	309
6.27	<code>activemq::core::ActiveMQConsumer</code> Class Reference . . . . .	310
6.27.1	Constructor & Destructor Documentation . . . . .	312
6.27.1.1	<code>ActiveMQConsumer</code> . . . . .	312
6.27.1.2	<code>~ActiveMQConsumer</code> . . . . .	312
6.27.2	Member Function Documentation . . . . .	312
6.27.2.1	<code>acknowledge</code> . . . . .	312
6.27.2.2	<code>acknowledge</code> . . . . .	313
6.27.2.3	<code>afterMessageIsConsumed</code> . . . . .	313
6.27.2.4	<code>beforeMessageIsConsumed</code> . . . . .	313
6.27.2.5	<code>clearMessagesInProgress</code> . . . . .	313
6.27.2.6	<code>close</code> . . . . .	313
6.27.2.7	<code>commit</code> . . . . .	314
6.27.2.8	<code>deliverAcks</code> . . . . .	314

6.27.2.9	dequeue . . . . .	314
6.27.2.10	dispatch . . . . .	314
6.27.2.11	doClose . . . . .	314
6.27.2.12	getConsumerId . . . . .	315
6.27.2.13	getConsumerInfo . . . . .	315
6.27.2.14	getLastDeliveredSequenceId . . . . .	315
6.27.2.15	getMessageAvailableCount . . . . .	315
6.27.2.16	getMessageListener . . . . .	315
6.27.2.17	getMessageSelector . . . . .	316
6.27.2.18	getRedeliveryPolicy . . . . .	316
6.27.2.19	inProgressClearRequired . . . . .	316
6.27.2.20	isClosed . . . . .	316
6.27.2.21	isSynchronizationRegistered . . . . .	316
6.27.2.22	iterate . . . . .	316
6.27.2.23	receive . . . . .	317
6.27.2.24	receive . . . . .	317
6.27.2.25	receiveNoWait . . . . .	317
6.27.2.26	rollback . . . . .	317
6.27.2.27	setLastDeliveredSequenceId . . . . .	318
6.27.2.28	setMessageListener . . . . .	318
6.27.2.29	setRedeliveryPolicy . . . . .	318
6.27.2.30	setSynchronizationRegistered . . . . .	318
6.27.2.31	start . . . . .	318
6.27.2.32	stop . . . . .	319
6.28	activemq::library::ActiveMQCPP Class Reference . . . . .	320
6.28.1	Constructor & Destructor Documentation . . . . .	320
6.28.1.1	ActiveMQCPP . . . . .	320
6.28.1.2	ActiveMQCPP . . . . .	320
6.28.1.3	~ActiveMQCPP . . . . .	320
6.28.2	Member Function Documentation . . . . .	320
6.28.2.1	initializeLibrary . . . . .	320
6.28.2.2	initializeLibrary . . . . .	321
6.28.2.3	operator= . . . . .	321
6.28.2.4	shutdownLibrary . . . . .	321
6.29	activemq::commands::ActiveMQDestination Class Reference . . . . .	322
6.29.1	Constructor & Destructor Documentation . . . . .	324

6.29.1.1	ActiveMQDestination . . . . .	324
6.29.1.2	ActiveMQDestination . . . . .	324
6.29.1.3	~ActiveMQDestination . . . . .	324
6.29.2	Member Function Documentation . . . . .	324
6.29.2.1	cloneDataStructure . . . . .	324
6.29.2.2	copyDataStructure . . . . .	325
6.29.2.3	createDestination . . . . .	325
6.29.2.4	createTemporaryName . . . . .	325
6.29.2.5	equals . . . . .	326
6.29.2.6	getClientId . . . . .	326
6.29.2.7	getCMSDestination . . . . .	326
6.29.2.8	getDataStructureType . . . . .	326
6.29.2.9	getDestinationType . . . . .	327
6.29.2.10	getOptions . . . . .	327
6.29.2.11	getOrderedTarget . . . . .	327
6.29.2.12	getPhysicalName . . . . .	327
6.29.2.13	getPhysicalName . . . . .	327
6.29.2.14	isAdvisory . . . . .	328
6.29.2.15	isComposite . . . . .	328
6.29.2.16	isConnectionAdvisory . . . . .	328
6.29.2.17	isConsumerAdvisory . . . . .	328
6.29.2.18	isExclusive . . . . .	328
6.29.2.19	isOrdered . . . . .	328
6.29.2.20	isProducerAdvisory . . . . .	329
6.29.2.21	isQueue . . . . .	329
6.29.2.22	isTemporary . . . . .	329
6.29.2.23	isTopic . . . . .	329
6.29.2.24	isWildcard . . . . .	329
6.29.2.25	setAdvisory . . . . .	329
6.29.2.26	setExclusive . . . . .	330
6.29.2.27	setOrdered . . . . .	330
6.29.2.28	setOrderedTarget . . . . .	330
6.29.2.29	setPhysicalName . . . . .	330
6.29.2.30	toString . . . . .	330
6.29.3	Field Documentation . . . . .	331
6.29.3.1	advisory . . . . .	331

6.29.3.2	ADVISORY_PREFIX . . . . .	331
6.29.3.3	COMPOSITE_SEPARATOR . . . . .	331
6.29.3.4	CONNECTION_ADVISORY_PREFIX . . . . .	331
6.29.3.5	CONSUMER_ADVISORY_PREFIX . . . . .	331
6.29.3.6	DEFAULT_ORDERED_TARGET . . . . .	331
6.29.3.7	exclusive . . . . .	332
6.29.3.8	ID_ACTIVEMQDESTINATION . . . . .	332
6.29.3.9	options . . . . .	332
6.29.3.10	ordered . . . . .	332
6.29.3.11	orderedTarget . . . . .	332
6.29.3.12	physicalName . . . . .	332
6.29.3.13	PRODUCER_ADVISORY_PREFIX . . . . .	332
6.29.3.14	QUEUE_QUALIFIED_PREFIX . . . . .	332
6.29.3.15	TEMP_POSTFIX . . . . .	332
6.29.3.16	TEMP_PREFIX . . . . .	332
6.29.3.17	TEMP_QUEUE_QUALIFIED_PREFIX . . . . .	332
6.29.3.18	TEMP_TOPIC_QUALIFIED_PREFIX . . . . .	332
6.29.3.19	TOPIC_QUALIFIED_PREFIX . . . . .	332
6.30	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	333
6.30.1	Detailed Description . . . . .	333
6.30.2	Constructor & Destructor Documentation . . . . .	334
6.30.2.1	ActiveMQDestinationMarshaller . . . . .	334
6.30.2.2	~ActiveMQDestinationMarshaller . . . . .	334
6.30.3	Member Function Documentation . . . . .	334
6.30.3.1	looseMarshal . . . . .	334
6.30.3.2	looseUnmarshal . . . . .	334
6.30.3.3	tightMarshal1 . . . . .	335
6.30.3.4	tightMarshal2 . . . . .	335
6.30.3.5	tightUnmarshal . . . . .	336
6.31	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	337
6.31.1	Detailed Description . . . . .	337
6.31.2	Constructor & Destructor Documentation . . . . .	338
6.31.2.1	ActiveMQDestinationMarshaller . . . . .	338
6.31.2.2	~ActiveMQDestinationMarshaller . . . . .	338
6.31.3	Member Function Documentation . . . . .	338



6.31.3.1	looseMarshal . . . . .	338
6.31.3.2	looseUnmarshal . . . . .	338
6.31.3.3	tightMarshal1 . . . . .	339
6.31.3.4	tightMarshal2 . . . . .	339
6.31.3.5	tightUnmarshal . . . . .	340
6.32	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	341
6.32.1	Detailed Description . . . . .	341
6.32.2	Constructor & Destructor Documentation . . . . .	342
6.32.2.1	ActiveMQDestinationMarshaller . . . . .	342
6.32.2.2	~ActiveMQDestinationMarshaller . . . . .	342
6.32.3	Member Function Documentation . . . . .	342
6.32.3.1	looseMarshal . . . . .	342
6.32.3.2	looseUnmarshal . . . . .	342
6.32.3.3	tightMarshal1 . . . . .	343
6.32.3.4	tightMarshal2 . . . . .	343
6.32.3.5	tightUnmarshal . . . . .	344
6.33	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	345
6.33.1	Detailed Description . . . . .	345
6.33.2	Constructor & Destructor Documentation . . . . .	346
6.33.2.1	ActiveMQDestinationMarshaller . . . . .	346
6.33.2.2	~ActiveMQDestinationMarshaller . . . . .	346
6.33.3	Member Function Documentation . . . . .	346
6.33.3.1	looseMarshal . . . . .	346
6.33.3.2	looseUnmarshal . . . . .	346
6.33.3.3	tightMarshal1 . . . . .	347
6.33.3.4	tightMarshal2 . . . . .	347
6.33.3.5	tightUnmarshal . . . . .	348
6.34	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	349
6.34.1	Detailed Description . . . . .	349
6.34.2	Constructor & Destructor Documentation . . . . .	350
6.34.2.1	ActiveMQDestinationMarshaller . . . . .	350
6.34.2.2	~ActiveMQDestinationMarshaller . . . . .	350
6.34.3	Member Function Documentation . . . . .	350
6.34.3.1	looseMarshal . . . . .	350

6.34.3.2	looseUnmarshal . . . . .	350
6.34.3.3	tightMarshal1 . . . . .	351
6.34.3.4	tightMarshal2 . . . . .	351
6.34.3.5	tightUnmarshal . . . . .	352
6.35	activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	353
6.35.1	Detailed Description . . . . .	353
6.35.2	Constructor & Destructor Documentation . . . . .	354
6.35.2.1	ActiveMQDestinationMarshaller . . . . .	354
6.35.2.2	~ActiveMQDestinationMarshaller . . . . .	354
6.35.3	Member Function Documentation . . . . .	354
6.35.3.1	looseMarshal . . . . .	354
6.35.3.2	looseUnmarshal . . . . .	354
6.35.3.3	tightMarshal1 . . . . .	355
6.35.3.4	tightMarshal2 . . . . .	355
6.35.3.5	tightUnmarshal . . . . .	356
6.36	activemq::exceptions::ActiveMQException Class Reference . . . . .	357
6.36.1	Constructor & Destructor Documentation . . . . .	357
6.36.1.1	ActiveMQException . . . . .	357
6.36.1.2	ActiveMQException . . . . .	357
6.36.1.3	ActiveMQException . . . . .	358
6.36.1.4	ActiveMQException . . . . .	358
6.36.1.5	~ActiveMQException . . . . .	358
6.36.2	Member Function Documentation . . . . .	358
6.36.2.1	clone . . . . .	358
6.36.2.2	convertToCMSException . . . . .	358
6.37	activemq::commands::ActiveMQMapMessage Class Reference . . . . .	360
6.37.1	Constructor & Destructor Documentation . . . . .	363
6.37.1.1	ActiveMQMapMessage . . . . .	363
6.37.1.2	~ActiveMQMapMessage . . . . .	363
6.37.2	Member Function Documentation . . . . .	363
6.37.2.1	beforeMarshal . . . . .	363
6.37.2.2	checkMapIsUnmarshalled . . . . .	363
6.37.2.3	clearBody . . . . .	363
6.37.2.4	clone . . . . .	363
6.37.2.5	cloneDataStructure . . . . .	364
6.37.2.6	copyDataStructure . . . . .	364

6.37.2.7	equals . . . . .	364
6.37.2.8	getBoolean . . . . .	364
6.37.2.9	getByte . . . . .	365
6.37.2.10	getBytes . . . . .	365
6.37.2.11	getChar . . . . .	365
6.37.2.12	getDataStructureType . . . . .	366
6.37.2.13	getDouble . . . . .	366
6.37.2.14	getFloat . . . . .	366
6.37.2.15	getInt . . . . .	366
6.37.2.16	getLong . . . . .	367
6.37.2.17	getMap . . . . .	367
6.37.2.18	getMap . . . . .	367
6.37.2.19	getMapNames . . . . .	367
6.37.2.20	getShort . . . . .	368
6.37.2.21	getString . . . . .	368
6.37.2.22	isMarshalAware . . . . .	368
6.37.2.23	itemExists . . . . .	368
6.37.2.24	setBoolean . . . . .	369
6.37.2.25	setByte . . . . .	369
6.37.2.26	setBytes . . . . .	370
6.37.2.27	setChar . . . . .	370
6.37.2.28	setDouble . . . . .	370
6.37.2.29	setFloat . . . . .	371
6.37.2.30	setInt . . . . .	371
6.37.2.31	setLong . . . . .	371
6.37.2.32	setShort . . . . .	372
6.37.2.33	setString . . . . .	372
6.37.2.34	toString . . . . .	372
6.37.3	Field Documentation . . . . .	372
6.37.3.1	ID_ACTIVEMQMAPMESSAGE . . . . .	372
6.38	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	374
6.38.1	Detailed Description . . . . .	374
6.38.2	Constructor & Destructor Documentation . . . . .	375
6.38.2.1	ActiveMQMapMessageMarshaller . . . . .	375
6.38.2.2	~ActiveMQMapMessageMarshaller . . . . .	375
6.38.3	Member Function Documentation . . . . .	375

6.38.3.1	createObject . . . . .	375
6.38.3.2	getDataStructureType . . . . .	375
6.38.3.3	looseMarshal . . . . .	375
6.38.3.4	looseUnmarshal . . . . .	376
6.38.3.5	tightMarshal1 . . . . .	376
6.38.3.6	tightMarshal2 . . . . .	376
6.38.3.7	tightUnmarshal . . . . .	377
6.39	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	378
6.39.1	Detailed Description . . . . .	378
6.39.2	Constructor & Destructor Documentation . . . . .	379
6.39.2.1	ActiveMQMapMessageMarshaller . . . . .	379
6.39.2.2	~ActiveMQMapMessageMarshaller . . . . .	379
6.39.3	Member Function Documentation . . . . .	379
6.39.3.1	createObject . . . . .	379
6.39.3.2	getDataStructureType . . . . .	379
6.39.3.3	looseMarshal . . . . .	379
6.39.3.4	looseUnmarshal . . . . .	380
6.39.3.5	tightMarshal1 . . . . .	380
6.39.3.6	tightMarshal2 . . . . .	380
6.39.3.7	tightUnmarshal . . . . .	381
6.40	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	382
6.40.1	Detailed Description . . . . .	382
6.40.2	Constructor & Destructor Documentation . . . . .	383
6.40.2.1	ActiveMQMapMessageMarshaller . . . . .	383
6.40.2.2	~ActiveMQMapMessageMarshaller . . . . .	383
6.40.3	Member Function Documentation . . . . .	383
6.40.3.1	createObject . . . . .	383
6.40.3.2	getDataStructureType . . . . .	383
6.40.3.3	looseMarshal . . . . .	383
6.40.3.4	looseUnmarshal . . . . .	384
6.40.3.5	tightMarshal1 . . . . .	384
6.40.3.6	tightMarshal2 . . . . .	384
6.40.3.7	tightUnmarshal . . . . .	385
6.41	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	386

6.41.1	Detailed Description . . . . .	386
6.41.2	Constructor & Destructor Documentation . . . . .	387
6.41.2.1	ActiveMQMapMessageMarshaller . . . . .	387
6.41.2.2	~ActiveMQMapMessageMarshaller . . . . .	387
6.41.3	Member Function Documentation . . . . .	387
6.41.3.1	createObject . . . . .	387
6.41.3.2	getDataStructureType . . . . .	387
6.41.3.3	looseMarshal . . . . .	387
6.41.3.4	looseUnmarshal . . . . .	388
6.41.3.5	tightMarshal1 . . . . .	388
6.41.3.6	tightMarshal2 . . . . .	388
6.41.3.7	tightUnmarshal . . . . .	389
6.42	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	390
6.42.1	Detailed Description . . . . .	390
6.42.2	Constructor & Destructor Documentation . . . . .	391
6.42.2.1	ActiveMQMapMessageMarshaller . . . . .	391
6.42.2.2	~ActiveMQMapMessageMarshaller . . . . .	391
6.42.3	Member Function Documentation . . . . .	391
6.42.3.1	createObject . . . . .	391
6.42.3.2	getDataStructureType . . . . .	391
6.42.3.3	looseMarshal . . . . .	391
6.42.3.4	looseUnmarshal . . . . .	392
6.42.3.5	tightMarshal1 . . . . .	392
6.42.3.6	tightMarshal2 . . . . .	392
6.42.3.7	tightUnmarshal . . . . .	393
6.43	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	394
6.43.1	Detailed Description . . . . .	394
6.43.2	Constructor & Destructor Documentation . . . . .	395
6.43.2.1	ActiveMQMapMessageMarshaller . . . . .	395
6.43.2.2	~ActiveMQMapMessageMarshaller . . . . .	395
6.43.3	Member Function Documentation . . . . .	395
6.43.3.1	createObject . . . . .	395
6.43.3.2	getDataStructureType . . . . .	395
6.43.3.3	looseMarshal . . . . .	395
6.43.3.4	looseUnmarshal . . . . .	396

6.43.3.5	tightMarshal1 . . . . .	396
6.43.3.6	tightMarshal2 . . . . .	396
6.43.3.7	tightUnmarshal . . . . .	397
6.44	activemq::commands::ActiveMQMessage Class Reference . . . . .	398
6.44.1	Constructor & Destructor Documentation . . . . .	398
6.44.1.1	ActiveMQMessage . . . . .	398
6.44.1.2	~ActiveMQMessage . . . . .	398
6.44.2	Member Function Documentation . . . . .	398
6.44.2.1	clone . . . . .	398
6.44.2.2	cloneDataStructure . . . . .	399
6.44.2.3	copyDataStructure . . . . .	399
6.44.2.4	equals . . . . .	399
6.44.2.5	getDataStructureType . . . . .	399
6.44.2.6	toString . . . . .	400
6.44.3	Field Documentation . . . . .	400
6.44.3.1	ID_ACTIVEMQMESSAGE . . . . .	400
6.45	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference . . . . .	401
6.45.1	Detailed Description . . . . .	401
6.45.2	Constructor & Destructor Documentation . . . . .	402
6.45.2.1	ActiveMQMessageMarshaller . . . . .	402
6.45.2.2	~ActiveMQMessageMarshaller . . . . .	402
6.45.3	Member Function Documentation . . . . .	402
6.45.3.1	createObject . . . . .	402
6.45.3.2	getDataStructureType . . . . .	402
6.45.3.3	looseMarshal . . . . .	402
6.45.3.4	looseUnmarshal . . . . .	403
6.45.3.5	tightMarshal1 . . . . .	403
6.45.3.6	tightMarshal2 . . . . .	403
6.45.3.7	tightUnmarshal . . . . .	404
6.46	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference . . . . .	405
6.46.1	Detailed Description . . . . .	405
6.46.2	Constructor & Destructor Documentation . . . . .	406
6.46.2.1	ActiveMQMessageMarshaller . . . . .	406
6.46.2.2	~ActiveMQMessageMarshaller . . . . .	406
6.46.3	Member Function Documentation . . . . .	406

6.46.3.1	createObject . . . . .	406
6.46.3.2	getDataStructureType . . . . .	406
6.46.3.3	looseMarshal . . . . .	406
6.46.3.4	looseUnmarshal . . . . .	407
6.46.3.5	tightMarshal1 . . . . .	407
6.46.3.6	tightMarshal2 . . . . .	407
6.46.3.7	tightUnmarshal . . . . .	408
6.47	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class	
	Reference . . . . .	409
6.47.1	Detailed Description . . . . .	409
6.47.2	Constructor & Destructor Documentation . . . . .	410
6.47.2.1	ActiveMQMessageMarshaller . . . . .	410
6.47.2.2	~ActiveMQMessageMarshaller . . . . .	410
6.47.3	Member Function Documentation . . . . .	410
6.47.3.1	createObject . . . . .	410
6.47.3.2	getDataStructureType . . . . .	410
6.47.3.3	looseMarshal . . . . .	410
6.47.3.4	looseUnmarshal . . . . .	411
6.47.3.5	tightMarshal1 . . . . .	411
6.47.3.6	tightMarshal2 . . . . .	411
6.47.3.7	tightUnmarshal . . . . .	412
6.48	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class	
	Reference . . . . .	413
6.48.1	Detailed Description . . . . .	413
6.48.2	Constructor & Destructor Documentation . . . . .	414
6.48.2.1	ActiveMQMessageMarshaller . . . . .	414
6.48.2.2	~ActiveMQMessageMarshaller . . . . .	414
6.48.3	Member Function Documentation . . . . .	414
6.48.3.1	createObject . . . . .	414
6.48.3.2	getDataStructureType . . . . .	414
6.48.3.3	looseMarshal . . . . .	414
6.48.3.4	looseUnmarshal . . . . .	415
6.48.3.5	tightMarshal1 . . . . .	415
6.48.3.6	tightMarshal2 . . . . .	415
6.48.3.7	tightUnmarshal . . . . .	416
6.49	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class	
	Reference . . . . .	417

6.49.1	Detailed Description . . . . .	417
6.49.2	Constructor & Destructor Documentation . . . . .	418
6.49.2.1	ActiveMQMessageMarshaller . . . . .	418
6.49.2.2	~ActiveMQMessageMarshaller . . . . .	418
6.49.3	Member Function Documentation . . . . .	418
6.49.3.1	createObject . . . . .	418
6.49.3.2	getDataStructureType . . . . .	418
6.49.3.3	looseMarshal . . . . .	418
6.49.3.4	looseUnmarshal . . . . .	419
6.49.3.5	tightMarshal1 . . . . .	419
6.49.3.6	tightMarshal2 . . . . .	419
6.49.3.7	tightUnmarshal . . . . .	420
6.50	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference . . . . .	421
6.50.1	Detailed Description . . . . .	421
6.50.2	Constructor & Destructor Documentation . . . . .	422
6.50.2.1	ActiveMQMessageMarshaller . . . . .	422
6.50.2.2	~ActiveMQMessageMarshaller . . . . .	422
6.50.3	Member Function Documentation . . . . .	422
6.50.3.1	createObject . . . . .	422
6.50.3.2	getDataStructureType . . . . .	422
6.50.3.3	looseMarshal . . . . .	422
6.50.3.4	looseUnmarshal . . . . .	423
6.50.3.5	tightMarshal1 . . . . .	423
6.50.3.6	tightMarshal2 . . . . .	423
6.50.3.7	tightUnmarshal . . . . .	424
6.51	activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference . . . . .	425
6.51.1	Constructor & Destructor Documentation . . . . .	428
6.51.1.1	ActiveMQMessageTemplate . . . . .	428
6.51.1.2	~ActiveMQMessageTemplate . . . . .	428
6.51.2	Member Function Documentation . . . . .	428
6.51.2.1	acknowledge . . . . .	428
6.51.2.2	clearBody . . . . .	428
6.51.2.3	clearProperties . . . . .	429
6.51.2.4	equals . . . . .	429
6.51.2.5	failIfReadOnlyBody . . . . .	429
6.51.2.6	failIfReadOnlyProperties . . . . .	429



6.51.2.7 failIfWriteOnlyBody . . . . .	429
6.51.2.8 getBooleanProperty . . . . .	429
6.51.2.9 getByteProperty . . . . .	430
6.51.2.10 getCMSCorrelationID . . . . .	430
6.51.2.11 getCMSDeliveryMode . . . . .	430
6.51.2.12 getCMSDestination . . . . .	431
6.51.2.13 getCMSExpiration . . . . .	431
6.51.2.14 getCMSMessageID . . . . .	431
6.51.2.15 getCMSPriority . . . . .	432
6.51.2.16 getCMSRedelivered . . . . .	432
6.51.2.17 getCMSReplyTo . . . . .	432
6.51.2.18 getCMSTimestamp . . . . .	432
6.51.2.19 getCMSType . . . . .	433
6.51.2.20 getDoubleProperty . . . . .	433
6.51.2.21 getFloatProperty . . . . .	433
6.51.2.22 getIntProperty . . . . .	434
6.51.2.23 getLongProperty . . . . .	434
6.51.2.24 getPropertyNames . . . . .	435
6.51.2.25 getShortProperty . . . . .	435
6.51.2.26 getStringProperty . . . . .	435
6.51.2.27 onSend . . . . .	436
6.51.2.28 propertyExists . . . . .	436
6.51.2.29 setBooleanProperty . . . . .	436
6.51.2.30 setByteProperty . . . . .	436
6.51.2.31 setCMSCorrelationID . . . . .	437
6.51.2.32 setCMSDeliveryMode . . . . .	437
6.51.2.33 setCMSDestination . . . . .	437
6.51.2.34 setCMSExpiration . . . . .	438
6.51.2.35 setCMSMessageID . . . . .	438
6.51.2.36 setCMSPriority . . . . .	438
6.51.2.37 setCMSRedelivered . . . . .	438
6.51.2.38 setCMSReplyTo . . . . .	439
6.51.2.39 setCMSTimestamp . . . . .	439
6.51.2.40 setCMSType . . . . .	439
6.51.2.41 setDoubleProperty . . . . .	440
6.51.2.42 setFloatProperty . . . . .	440

6.51.2.43	setIntProperty . . . . .	440
6.51.2.44	setLongProperty . . . . .	441
6.51.2.45	setShortProperty . . . . .	441
6.51.2.46	setStringProperty . . . . .	441
6.52	activemq::commands::ActiveMQObjectMessage Class Reference . . . . .	443
6.52.1	Constructor & Destructor Documentation . . . . .	444
6.52.1.1	ActiveMQObjectMessage . . . . .	444
6.52.1.2	~ActiveMQObjectMessage . . . . .	444
6.52.2	Member Function Documentation . . . . .	444
6.52.2.1	clone . . . . .	444
6.52.2.2	cloneDataStructure . . . . .	444
6.52.2.3	copyDataStructure . . . . .	444
6.52.2.4	equals . . . . .	444
6.52.2.5	getDataStructureType . . . . .	445
6.52.2.6	toString . . . . .	445
6.52.3	Field Documentation . . . . .	445
6.52.3.1	ID_ACTIVEMQOBJECTMESSAGE . . . . .	445
6.53	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference . . . . .	446
6.53.1	Detailed Description . . . . .	446
6.53.2	Constructor & Destructor Documentation . . . . .	447
6.53.2.1	ActiveMQObjectMessageMarshaller . . . . .	447
6.53.2.2	~ActiveMQObjectMessageMarshaller . . . . .	447
6.53.3	Member Function Documentation . . . . .	447
6.53.3.1	createObject . . . . .	447
6.53.3.2	getDataStructureType . . . . .	447
6.53.3.3	looseMarshal . . . . .	447
6.53.3.4	looseUnmarshal . . . . .	448
6.53.3.5	tightMarshal1 . . . . .	448
6.53.3.6	tightMarshal2 . . . . .	448
6.53.3.7	tightUnmarshal . . . . .	449
6.54	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference . . . . .	450
6.54.1	Detailed Description . . . . .	450
6.54.2	Constructor & Destructor Documentation . . . . .	451
6.54.2.1	ActiveMQObjectMessageMarshaller . . . . .	451
6.54.2.2	~ActiveMQObjectMessageMarshaller . . . . .	451

6.54.3	Member Function Documentation . . . . .	451
6.54.3.1	createObject . . . . .	451
6.54.3.2	getDataStructureType . . . . .	451
6.54.3.3	looseMarshal . . . . .	451
6.54.3.4	looseUnmarshal . . . . .	452
6.54.3.5	tightMarshal1 . . . . .	452
6.54.3.6	tightMarshal2 . . . . .	452
6.54.3.7	tightUnmarshal . . . . .	453
6.55	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference . . . . .	454
6.55.1	Detailed Description . . . . .	454
6.55.2	Constructor & Destructor Documentation . . . . .	455
6.55.2.1	ActiveMQObjectMessageMarshaller . . . . .	455
6.55.2.2	~ActiveMQObjectMessageMarshaller . . . . .	455
6.55.3	Member Function Documentation . . . . .	455
6.55.3.1	createObject . . . . .	455
6.55.3.2	getDataStructureType . . . . .	455
6.55.3.3	looseMarshal . . . . .	455
6.55.3.4	looseUnmarshal . . . . .	456
6.55.3.5	tightMarshal1 . . . . .	456
6.55.3.6	tightMarshal2 . . . . .	456
6.55.3.7	tightUnmarshal . . . . .	457
6.56	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference . . . . .	458
6.56.1	Detailed Description . . . . .	458
6.56.2	Constructor & Destructor Documentation . . . . .	459
6.56.2.1	ActiveMQObjectMessageMarshaller . . . . .	459
6.56.2.2	~ActiveMQObjectMessageMarshaller . . . . .	459
6.56.3	Member Function Documentation . . . . .	459
6.56.3.1	createObject . . . . .	459
6.56.3.2	getDataStructureType . . . . .	459
6.56.3.3	looseMarshal . . . . .	459
6.56.3.4	looseUnmarshal . . . . .	460
6.56.3.5	tightMarshal1 . . . . .	460
6.56.3.6	tightMarshal2 . . . . .	460
6.56.3.7	tightUnmarshal . . . . .	461

6.57	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	462
6.57.1	Detailed Description . . . . .	462
6.57.2	Constructor & Destructor Documentation . . . . .	463
	6.57.2.1 ActiveMQObjectMessageMarshaller . . . . .	463
	6.57.2.2 ~ActiveMQObjectMessageMarshaller . . . . .	463
6.57.3	Member Function Documentation . . . . .	463
	6.57.3.1 createObject . . . . .	463
	6.57.3.2 getDataStructureType . . . . .	463
	6.57.3.3 looseMarshal . . . . .	463
	6.57.3.4 looseUnmarshal . . . . .	464
	6.57.3.5 tightMarshal1 . . . . .	464
	6.57.3.6 tightMarshal2 . . . . .	464
	6.57.3.7 tightUnmarshal . . . . .	465
6.58	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	466
6.58.1	Detailed Description . . . . .	466
6.58.2	Constructor & Destructor Documentation . . . . .	467
	6.58.2.1 ActiveMQObjectMessageMarshaller . . . . .	467
	6.58.2.2 ~ActiveMQObjectMessageMarshaller . . . . .	467
6.58.3	Member Function Documentation . . . . .	467
	6.58.3.1 createObject . . . . .	467
	6.58.3.2 getDataStructureType . . . . .	467
	6.58.3.3 looseMarshal . . . . .	467
	6.58.3.4 looseUnmarshal . . . . .	468
	6.58.3.5 tightMarshal1 . . . . .	468
	6.58.3.6 tightMarshal2 . . . . .	468
	6.58.3.7 tightUnmarshal . . . . .	469
6.59	activemq::core::ActiveMQProducer Class Reference . . . . .	470
6.59.1	Constructor & Destructor Documentation . . . . .	471
	6.59.1.1 ActiveMQProducer . . . . .	471
	6.59.1.2 ~ActiveMQProducer . . . . .	472
6.59.2	Member Function Documentation . . . . .	472
	6.59.2.1 close . . . . .	472
	6.59.2.2 getDeliveryMode . . . . .	472
	6.59.2.3 getDisableMessageID . . . . .	472
	6.59.2.4 getDisableMessageTimeStamp . . . . .	472

6.59.2.5	getPriority . . . . .	473
6.59.2.6	getProducerId . . . . .	473
6.59.2.7	getProducerInfo . . . . .	473
6.59.2.8	getSendTimeout . . . . .	473
6.59.2.9	getTimeToLive . . . . .	473
6.59.2.10	isClosed . . . . .	474
6.59.2.11	onProducerAck . . . . .	474
6.59.2.12	send . . . . .	474
6.59.2.13	send . . . . .	475
6.59.2.14	send . . . . .	475
6.59.2.15	send . . . . .	476
6.59.2.16	setDeliveryMode . . . . .	476
6.59.2.17	setDisableMessageID . . . . .	476
6.59.2.18	setDisableMessageTimeStamp . . . . .	476
6.59.2.19	setPriority . . . . .	477
6.59.2.20	setSendTimeout . . . . .	477
6.59.2.21	setTimeToLive . . . . .	477
6.60	activemq::util::ActiveMQProperties Class Reference . . . . .	478
6.60.1	Detailed Description . . . . .	479
6.60.2	Constructor & Destructor Documentation . . . . .	479
6.60.2.1	ActiveMQProperties . . . . .	479
6.60.2.2	~ActiveMQProperties . . . . .	479
6.60.3	Member Function Documentation . . . . .	479
6.60.3.1	clear . . . . .	479
6.60.3.2	clone . . . . .	479
6.60.3.3	copy . . . . .	479
6.60.3.4	getProperties . . . . .	480
6.60.3.5	getProperties . . . . .	480
6.60.3.6	getProperty . . . . .	480
6.60.3.7	getProperty . . . . .	480
6.60.3.8	hasProperty . . . . .	480
6.60.3.9	isEmpty . . . . .	481
6.60.3.10	remove . . . . .	481
6.60.3.11	setProperties . . . . .	481
6.60.3.12	setProperty . . . . .	481
6.60.3.13	toArray . . . . .	481

6.60.3.14	toString . . . . .	481
6.61	activemq::commands::ActiveMQQueue Class Reference . . . . .	483
6.61.1	Constructor & Destructor Documentation . . . . .	484
6.61.1.1	ActiveMQQueue . . . . .	484
6.61.1.2	ActiveMQQueue . . . . .	484
6.61.1.3	~ActiveMQQueue . . . . .	484
6.61.2	Member Function Documentation . . . . .	484
6.61.2.1	clone . . . . .	484
6.61.2.2	cloneDataStructure . . . . .	484
6.61.2.3	copy . . . . .	484
6.61.2.4	copyDataStructure . . . . .	484
6.61.2.5	equals . . . . .	485
6.61.2.6	getCMSDestination . . . . .	485
6.61.2.7	getCMSProperties . . . . .	485
6.61.2.8	getDataStructureType . . . . .	485
6.61.2.9	getDestinationType . . . . .	485
6.61.2.10	getQueueName . . . . .	486
6.61.2.11	toString . . . . .	486
6.61.3	Field Documentation . . . . .	486
6.61.3.1	ID_ACTIVEMQQUEUE . . . . .	486
6.62	activemq::core::ActiveMQQueueBrowser Class Reference . . . . .	487
6.62.1	Constructor & Destructor Documentation . . . . .	488
6.62.1.1	ActiveMQQueueBrowser . . . . .	488
6.62.1.2	~ActiveMQQueueBrowser . . . . .	488
6.62.2	Member Function Documentation . . . . .	488
6.62.2.1	close . . . . .	488
6.62.2.2	getEnumeration . . . . .	488
6.62.2.3	getMessageSelector . . . . .	488
6.62.2.4	getQueue . . . . .	489
6.62.2.5	hasMoreMessages . . . . .	489
6.62.2.6	nextMessage . . . . .	489
6.62.3	Friends And Related Function Documentation . . . . .	490
6.62.3.1	Browser . . . . .	490
6.63	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference . . . . .	491
6.63.1	Detailed Description . . . . .	491
6.63.2	Constructor & Destructor Documentation . . . . .	492

6.63.2.1	ActiveMQQueueMarshaller . . . . .	492
6.63.2.2	~ActiveMQQueueMarshaller . . . . .	492
6.63.3	Member Function Documentation . . . . .	492
6.63.3.1	createObject . . . . .	492
6.63.3.2	getDataStructureType . . . . .	492
6.63.3.3	looseMarshal . . . . .	492
6.63.3.4	looseUnmarshal . . . . .	493
6.63.3.5	tightMarshal1 . . . . .	493
6.63.3.6	tightMarshal2 . . . . .	493
6.63.3.7	tightUnmarshal . . . . .	494
6.64	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class	
	Reference . . . . .	495
6.64.1	Detailed Description . . . . .	495
6.64.2	Constructor & Destructor Documentation . . . . .	496
6.64.2.1	ActiveMQQueueMarshaller . . . . .	496
6.64.2.2	~ActiveMQQueueMarshaller . . . . .	496
6.64.3	Member Function Documentation . . . . .	496
6.64.3.1	createObject . . . . .	496
6.64.3.2	getDataStructureType . . . . .	496
6.64.3.3	looseMarshal . . . . .	496
6.64.3.4	looseUnmarshal . . . . .	497
6.64.3.5	tightMarshal1 . . . . .	497
6.64.3.6	tightMarshal2 . . . . .	497
6.64.3.7	tightUnmarshal . . . . .	498
6.65	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class	
	Reference . . . . .	499
6.65.1	Detailed Description . . . . .	499
6.65.2	Constructor & Destructor Documentation . . . . .	500
6.65.2.1	ActiveMQQueueMarshaller . . . . .	500
6.65.2.2	~ActiveMQQueueMarshaller . . . . .	500
6.65.3	Member Function Documentation . . . . .	500
6.65.3.1	createObject . . . . .	500
6.65.3.2	getDataStructureType . . . . .	500
6.65.3.3	looseMarshal . . . . .	500
6.65.3.4	looseUnmarshal . . . . .	501
6.65.3.5	tightMarshal1 . . . . .	501
6.65.3.6	tightMarshal2 . . . . .	501

6.65.3.7	tightUnmarshal	502
6.66	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	Class
	Reference	503
6.66.1	Detailed Description	503
6.66.2	Constructor & Destructor Documentation	504
6.66.2.1	ActiveMQQueueMarshaller	504
6.66.2.2	~ActiveMQQueueMarshaller	504
6.66.3	Member Function Documentation	504
6.66.3.1	createObject	504
6.66.3.2	getDataStructureType	504
6.66.3.3	looseMarshal	504
6.66.3.4	looseUnmarshal	505
6.66.3.5	tightMarshal1	505
6.66.3.6	tightMarshal2	505
6.66.3.7	tightUnmarshal	506
6.67	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	Class
	Reference	507
6.67.1	Detailed Description	507
6.67.2	Constructor & Destructor Documentation	508
6.67.2.1	ActiveMQQueueMarshaller	508
6.67.2.2	~ActiveMQQueueMarshaller	508
6.67.3	Member Function Documentation	508
6.67.3.1	createObject	508
6.67.3.2	getDataStructureType	508
6.67.3.3	looseMarshal	508
6.67.3.4	looseUnmarshal	509
6.67.3.5	tightMarshal1	509
6.67.3.6	tightMarshal2	509
6.67.3.7	tightUnmarshal	510
6.68	activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	Class
	Reference	511
6.68.1	Detailed Description	511
6.68.2	Constructor & Destructor Documentation	512
6.68.2.1	ActiveMQQueueMarshaller	512
6.68.2.2	~ActiveMQQueueMarshaller	512
6.68.3	Member Function Documentation	512
6.68.3.1	createObject	512



6.68.3.2	getDataStructureType . . . . .	512
6.68.3.3	looseMarshal . . . . .	512
6.68.3.4	looseUnmarshal . . . . .	513
6.68.3.5	tightMarshal1 . . . . .	513
6.68.3.6	tightMarshal2 . . . . .	513
6.68.3.7	tightUnmarshal . . . . .	514
6.69	activemq::core::ActiveMQSession Class Reference . . . . .	515
6.69.1	Constructor & Destructor Documentation . . . . .	519
6.69.1.1	ActiveMQSession . . . . .	519
6.69.1.2	~ActiveMQSession . . . . .	519
6.69.2	Member Function Documentation . . . . .	519
6.69.2.1	acknowledge . . . . .	519
6.69.2.2	addConsumer . . . . .	519
6.69.2.3	addProducer . . . . .	519
6.69.2.4	clearMessagesInProgress . . . . .	520
6.69.2.5	close . . . . .	520
6.69.2.6	commit . . . . .	520
6.69.2.7	createBrowser . . . . .	520
6.69.2.8	createBrowser . . . . .	521
6.69.2.9	createBytesMessage . . . . .	521
6.69.2.10	createBytesMessage . . . . .	521
6.69.2.11	createConsumer . . . . .	522
6.69.2.12	createConsumer . . . . .	522
6.69.2.13	createConsumer . . . . .	522
6.69.2.14	createDurableConsumer . . . . .	523
6.69.2.15	createMapMessage . . . . .	523
6.69.2.16	createMessage . . . . .	523
6.69.2.17	createProducer . . . . .	524
6.69.2.18	createQueue . . . . .	524
6.69.2.19	createStreamMessage . . . . .	524
6.69.2.20	createTemporaryQueue . . . . .	524
6.69.2.21	createTemporaryTopic . . . . .	525
6.69.2.22	createTextMessage . . . . .	525
6.69.2.23	createTextMessage . . . . .	525
6.69.2.24	createTopic . . . . .	526
6.69.2.25	deliverAcks . . . . .	526

6.69.2.26	dispatch . . . . .	526
6.69.2.27	doStartTransaction . . . . .	526
6.69.2.28	fire . . . . .	526
6.69.2.29	getAcknowledgeMode . . . . .	526
6.69.2.30	getConnection . . . . .	527
6.69.2.31	getExceptionListener . . . . .	527
6.69.2.32	getLastDeliveredSequenceId . . . . .	527
6.69.2.33	getNextConsumerId . . . . .	527
6.69.2.34	getNextProducerId . . . . .	527
6.69.2.35	getSessionId . . . . .	528
6.69.2.36	getSessionInfo . . . . .	528
6.69.2.37	getTransactionContext . . . . .	528
6.69.2.38	isAutoAcknowledge . . . . .	528
6.69.2.39	isClientAcknowledge . . . . .	528
6.69.2.40	isDupsOkAcknowledge . . . . .	528
6.69.2.41	isIndividualAcknowledge . . . . .	528
6.69.2.42	isStarted . . . . .	529
6.69.2.43	isTransacted . . . . .	529
6.69.2.44	oneway . . . . .	529
6.69.2.45	recover . . . . .	529
6.69.2.46	redispatch . . . . .	530
6.69.2.47	removeConsumer . . . . .	530
6.69.2.48	removeProducer . . . . .	530
6.69.2.49	rollback . . . . .	530
6.69.2.50	send . . . . .	531
6.69.2.51	setLastDeliveredSequenceId . . . . .	531
6.69.2.52	start . . . . .	531
6.69.2.53	stop . . . . .	531
6.69.2.54	syncRequest . . . . .	531
6.69.2.55	unsubscribe . . . . .	532
6.69.2.56	wakeup . . . . .	532
6.69.3	Friends And Related Function Documentation . . . . .	532
6.69.3.1	ActiveMQSessionExecutor . . . . .	532
6.70	activemq::core::ActiveMQSessionExecutor Class Reference . . . . .	533
6.70.1	Detailed Description . . . . .	534
6.70.2	Constructor & Destructor Documentation . . . . .	534

6.70.2.1	ActiveMQSessionExecutor . . . . .	534
6.70.2.2	~ActiveMQSessionExecutor . . . . .	534
6.70.3	Member Function Documentation . . . . .	534
6.70.3.1	clear . . . . .	534
6.70.3.2	clearMessagesInProgress . . . . .	534
6.70.3.3	close . . . . .	534
6.70.3.4	execute . . . . .	534
6.70.3.5	executeFirst . . . . .	534
6.70.3.6	getUnconsumedMessages . . . . .	535
6.70.3.7	hasUnconsumedMessages . . . . .	535
6.70.3.8	isEmpty . . . . .	535
6.70.3.9	isRunning . . . . .	535
6.70.3.10	iterate . . . . .	535
6.70.3.11	start . . . . .	535
6.70.3.12	stop . . . . .	536
6.70.3.13	wakeup . . . . .	536
6.71	activemq::commands::ActiveMQStreamMessage Class Reference . . . . .	537
6.71.1	Constructor & Destructor Documentation . . . . .	540
6.71.1.1	ActiveMQStreamMessage . . . . .	540
6.71.1.2	~ActiveMQStreamMessage . . . . .	540
6.71.2	Member Function Documentation . . . . .	540
6.71.2.1	clearBody . . . . .	540
6.71.2.2	clone . . . . .	540
6.71.2.3	cloneDataStructure . . . . .	540
6.71.2.4	copyDataStructure . . . . .	540
6.71.2.5	equals . . . . .	541
6.71.2.6	getDataStructureType . . . . .	541
6.71.2.7	onSend . . . . .	541
6.71.2.8	readBoolean . . . . .	541
6.71.2.9	readByte . . . . .	542
6.71.2.10	readBytes . . . . .	542
6.71.2.11	readBytes . . . . .	543
6.71.2.12	readChar . . . . .	543
6.71.2.13	readDouble . . . . .	544
6.71.2.14	readFloat . . . . .	544
6.71.2.15	readInt . . . . .	545

6.71.2.16	readLong . . . . .	545
6.71.2.17	readShort . . . . .	545
6.71.2.18	readString . . . . .	546
6.71.2.19	readUnsignedShort . . . . .	546
6.71.2.20	reset . . . . .	547
6.71.2.21	toString . . . . .	547
6.71.2.22	writeBoolean . . . . .	547
6.71.2.23	writeByte . . . . .	547
6.71.2.24	writeBytes . . . . .	548
6.71.2.25	writeBytes . . . . .	548
6.71.2.26	writeChar . . . . .	548
6.71.2.27	writeDouble . . . . .	549
6.71.2.28	writeFloat . . . . .	549
6.71.2.29	writeInt . . . . .	549
6.71.2.30	writeLong . . . . .	550
6.71.2.31	writeShort . . . . .	550
6.71.2.32	writeString . . . . .	550
6.71.2.33	writeUnsignedShort . . . . .	551
6.71.3	Field Documentation . . . . .	551
6.71.3.1	ID_ ACTIVEMQSTREAMMESSAGE . . . . .	551
6.72	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	552
6.72.1	Detailed Description . . . . .	552
6.72.2	Constructor & Destructor Documentation . . . . .	553
6.72.2.1	ActiveMQStreamMessageMarshaller . . . . .	553
6.72.2.2	~ActiveMQStreamMessageMarshaller . . . . .	553
6.72.3	Member Function Documentation . . . . .	553
6.72.3.1	createObject . . . . .	553
6.72.3.2	getDataStructureType . . . . .	553
6.72.3.3	looseMarshal . . . . .	553
6.72.3.4	looseUnmarshal . . . . .	554
6.72.3.5	tightMarshal1 . . . . .	554
6.72.3.6	tightMarshal2 . . . . .	554
6.72.3.7	tightUnmarshal . . . . .	555
6.73	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	556
6.73.1	Detailed Description . . . . .	556

6.73.2	Constructor & Destructor Documentation . . . . .	557
6.73.2.1	ActiveMQStreamMessageMarshaller . . . . .	557
6.73.2.2	~ActiveMQStreamMessageMarshaller . . . . .	557
6.73.3	Member Function Documentation . . . . .	557
6.73.3.1	createObject . . . . .	557
6.73.3.2	getDataStructureType . . . . .	557
6.73.3.3	looseMarshal . . . . .	557
6.73.3.4	looseUnmarshal . . . . .	558
6.73.3.5	tightMarshal1 . . . . .	558
6.73.3.6	tightMarshal2 . . . . .	558
6.73.3.7	tightUnmarshal . . . . .	559
6.74	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	560
6.74.1	Detailed Description . . . . .	560
6.74.2	Constructor & Destructor Documentation . . . . .	561
6.74.2.1	ActiveMQStreamMessageMarshaller . . . . .	561
6.74.2.2	~ActiveMQStreamMessageMarshaller . . . . .	561
6.74.3	Member Function Documentation . . . . .	561
6.74.3.1	createObject . . . . .	561
6.74.3.2	getDataStructureType . . . . .	561
6.74.3.3	looseMarshal . . . . .	561
6.74.3.4	looseUnmarshal . . . . .	562
6.74.3.5	tightMarshal1 . . . . .	562
6.74.3.6	tightMarshal2 . . . . .	562
6.74.3.7	tightUnmarshal . . . . .	563
6.75	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	564
6.75.1	Detailed Description . . . . .	564
6.75.2	Constructor & Destructor Documentation . . . . .	565
6.75.2.1	ActiveMQStreamMessageMarshaller . . . . .	565
6.75.2.2	~ActiveMQStreamMessageMarshaller . . . . .	565
6.75.3	Member Function Documentation . . . . .	565
6.75.3.1	createObject . . . . .	565
6.75.3.2	getDataStructureType . . . . .	565
6.75.3.3	looseMarshal . . . . .	565
6.75.3.4	looseUnmarshal . . . . .	566
6.75.3.5	tightMarshal1 . . . . .	566

6.75.3.6	tightMarshal2 . . . . .	566
6.75.3.7	tightUnmarshal . . . . .	567
6.76	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	568
6.76.1	Detailed Description . . . . .	568
6.76.2	Constructor & Destructor Documentation . . . . .	569
6.76.2.1	ActiveMQStreamMessageMarshaller . . . . .	569
6.76.2.2	~ActiveMQStreamMessageMarshaller . . . . .	569
6.76.3	Member Function Documentation . . . . .	569
6.76.3.1	createObject . . . . .	569
6.76.3.2	getDataStructureType . . . . .	569
6.76.3.3	looseMarshal . . . . .	569
6.76.3.4	looseUnmarshal . . . . .	570
6.76.3.5	tightMarshal1 . . . . .	570
6.76.3.6	tightMarshal2 . . . . .	570
6.76.3.7	tightUnmarshal . . . . .	571
6.77	activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	572
6.77.1	Detailed Description . . . . .	572
6.77.2	Constructor & Destructor Documentation . . . . .	573
6.77.2.1	ActiveMQStreamMessageMarshaller . . . . .	573
6.77.2.2	~ActiveMQStreamMessageMarshaller . . . . .	573
6.77.3	Member Function Documentation . . . . .	573
6.77.3.1	createObject . . . . .	573
6.77.3.2	getDataStructureType . . . . .	573
6.77.3.3	looseMarshal . . . . .	573
6.77.3.4	looseUnmarshal . . . . .	574
6.77.3.5	tightMarshal1 . . . . .	574
6.77.3.6	tightMarshal2 . . . . .	574
6.77.3.7	tightUnmarshal . . . . .	575
6.78	activemq::commands::ActiveMQTempDestination Class Reference . . . . .	576
6.78.1	Constructor & Destructor Documentation . . . . .	577
6.78.1.1	ActiveMQTempDestination . . . . .	577
6.78.1.2	ActiveMQTempDestination . . . . .	577
6.78.1.3	~ActiveMQTempDestination . . . . .	577
6.78.2	Member Function Documentation . . . . .	577
6.78.2.1	cloneDataStructure . . . . .	577

6.78.2.2	close . . . . .	577
6.78.2.3	copyDataStructure . . . . .	577
6.78.2.4	equals . . . . .	578
6.78.2.5	getDataStructureType . . . . .	578
6.78.2.6	setConnection . . . . .	578
6.78.2.7	toString . . . . .	578
6.78.3	Field Documentation . . . . .	579
6.78.3.1	connection . . . . .	579
6.78.3.2	ID_ACTIVEMQTEMPDESTINATION . . . . .	579
6.79	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	580
6.79.1	Detailed Description . . . . .	580
6.79.2	Constructor & Destructor Documentation . . . . .	581
6.79.2.1	ActiveMQTempDestinationMarshaller . . . . .	581
6.79.2.2	~ActiveMQTempDestinationMarshaller . . . . .	581
6.79.3	Member Function Documentation . . . . .	581
6.79.3.1	looseMarshal . . . . .	581
6.79.3.2	looseUnmarshal . . . . .	581
6.79.3.3	tightMarshal1 . . . . .	582
6.79.3.4	tightMarshal2 . . . . .	582
6.79.3.5	tightUnmarshal . . . . .	583
6.80	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	584
6.80.1	Detailed Description . . . . .	584
6.80.2	Constructor & Destructor Documentation . . . . .	585
6.80.2.1	ActiveMQTempDestinationMarshaller . . . . .	585
6.80.2.2	~ActiveMQTempDestinationMarshaller . . . . .	585
6.80.3	Member Function Documentation . . . . .	585
6.80.3.1	looseMarshal . . . . .	585
6.80.3.2	looseUnmarshal . . . . .	585
6.80.3.3	tightMarshal1 . . . . .	586
6.80.3.4	tightMarshal2 . . . . .	586
6.80.3.5	tightUnmarshal . . . . .	587
6.81	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	588
6.81.1	Detailed Description . . . . .	588
6.81.2	Constructor & Destructor Documentation . . . . .	589

6.81.2.1	ActiveMQTempDestinationMarshaller . . . . .	589
6.81.2.2	~ActiveMQTempDestinationMarshaller . . . . .	589
6.81.3	Member Function Documentation . . . . .	589
6.81.3.1	looseMarshal . . . . .	589
6.81.3.2	looseUnmarshal . . . . .	589
6.81.3.3	tightMarshal1 . . . . .	590
6.81.3.4	tightMarshal2 . . . . .	590
6.81.3.5	tightUnmarshal . . . . .	591
6.82	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference . . . . .	592
6.82.1	Detailed Description . . . . .	592
6.82.2	Constructor & Destructor Documentation . . . . .	593
6.82.2.1	ActiveMQTempDestinationMarshaller . . . . .	593
6.82.2.2	~ActiveMQTempDestinationMarshaller . . . . .	593
6.82.3	Member Function Documentation . . . . .	593
6.82.3.1	looseMarshal . . . . .	593
6.82.3.2	looseUnmarshal . . . . .	593
6.82.3.3	tightMarshal1 . . . . .	594
6.82.3.4	tightMarshal2 . . . . .	594
6.82.3.5	tightUnmarshal . . . . .	595
6.83	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference . . . . .	596
6.83.1	Detailed Description . . . . .	596
6.83.2	Constructor & Destructor Documentation . . . . .	597
6.83.2.1	ActiveMQTempDestinationMarshaller . . . . .	597
6.83.2.2	~ActiveMQTempDestinationMarshaller . . . . .	597
6.83.3	Member Function Documentation . . . . .	597
6.83.3.1	looseMarshal . . . . .	597
6.83.3.2	looseUnmarshal . . . . .	597
6.83.3.3	tightMarshal1 . . . . .	598
6.83.3.4	tightMarshal2 . . . . .	598
6.83.3.5	tightUnmarshal . . . . .	599
6.84	activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller Class Reference . . . . .	600
6.84.1	Detailed Description . . . . .	600
6.84.2	Constructor & Destructor Documentation . . . . .	601
6.84.2.1	ActiveMQTempDestinationMarshaller . . . . .	601



6.84.2.2	<code>~ActiveMQTempDestinationMarshaller</code> . . . . .	601
6.84.3	Member Function Documentation . . . . .	601
6.84.3.1	<code>looseMarshal</code> . . . . .	601
6.84.3.2	<code>looseUnmarshal</code> . . . . .	601
6.84.3.3	<code>tightMarshal1</code> . . . . .	602
6.84.3.4	<code>tightMarshal2</code> . . . . .	602
6.84.3.5	<code>tightUnmarshal</code> . . . . .	603
6.85	<code>activemq::commands::ActiveMQTempQueue</code> Class Reference . . . . .	604
6.85.1	Constructor & Destructor Documentation . . . . .	605
6.85.1.1	<code>ActiveMQTempQueue</code> . . . . .	605
6.85.1.2	<code>ActiveMQTempQueue</code> . . . . .	605
6.85.1.3	<code>~ActiveMQTempQueue</code> . . . . .	605
6.85.2	Member Function Documentation . . . . .	605
6.85.2.1	<code>clone</code> . . . . .	605
6.85.2.2	<code>cloneDataStructure</code> . . . . .	605
6.85.2.3	<code>copy</code> . . . . .	605
6.85.2.4	<code>copyDataStructure</code> . . . . .	606
6.85.2.5	<code>destroy</code> . . . . .	606
6.85.2.6	<code>equals</code> . . . . .	606
6.85.2.7	<code>getCMSDestination</code> . . . . .	606
6.85.2.8	<code>getCMSProperties</code> . . . . .	606
6.85.2.9	<code>getDataStructureType</code> . . . . .	607
6.85.2.10	<code>getDestinationType</code> . . . . .	607
6.85.2.11	<code>getQueueName</code> . . . . .	607
6.85.2.12	<code>toString</code> . . . . .	607
6.85.3	Field Documentation . . . . .	608
6.85.3.1	<code>ID_ACTIVEMQTEMPQUEUE</code> . . . . .	608
6.86	<code>activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller</code> Class Reference . . . . .	609
6.86.1	Detailed Description . . . . .	609
6.86.2	Constructor & Destructor Documentation . . . . .	610
6.86.2.1	<code>ActiveMQTempQueueMarshaller</code> . . . . .	610
6.86.2.2	<code>~ActiveMQTempQueueMarshaller</code> . . . . .	610
6.86.3	Member Function Documentation . . . . .	610
6.86.3.1	<code>createObject</code> . . . . .	610
6.86.3.2	<code>getDataStructureType</code> . . . . .	610
6.86.3.3	<code>looseMarshal</code> . . . . .	610

6.86.3.4	looseUnmarshal . . . . .	611
6.86.3.5	tightMarshal1 . . . . .	611
6.86.3.6	tightMarshal2 . . . . .	611
6.86.3.7	tightUnmarshal . . . . .	612
6.87	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	613
6.87.1	Detailed Description . . . . .	613
6.87.2	Constructor & Destructor Documentation . . . . .	614
6.87.2.1	ActiveMQTempQueueMarshaller . . . . .	614
6.87.2.2	~ActiveMQTempQueueMarshaller . . . . .	614
6.87.3	Member Function Documentation . . . . .	614
6.87.3.1	createObject . . . . .	614
6.87.3.2	getDataStructureType . . . . .	614
6.87.3.3	looseMarshal . . . . .	614
6.87.3.4	looseUnmarshal . . . . .	615
6.87.3.5	tightMarshal1 . . . . .	615
6.87.3.6	tightMarshal2 . . . . .	615
6.87.3.7	tightUnmarshal . . . . .	616
6.88	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	617
6.88.1	Detailed Description . . . . .	617
6.88.2	Constructor & Destructor Documentation . . . . .	618
6.88.2.1	ActiveMQTempQueueMarshaller . . . . .	618
6.88.2.2	~ActiveMQTempQueueMarshaller . . . . .	618
6.88.3	Member Function Documentation . . . . .	618
6.88.3.1	createObject . . . . .	618
6.88.3.2	getDataStructureType . . . . .	618
6.88.3.3	looseMarshal . . . . .	618
6.88.3.4	looseUnmarshal . . . . .	619
6.88.3.5	tightMarshal1 . . . . .	619
6.88.3.6	tightMarshal2 . . . . .	619
6.88.3.7	tightUnmarshal . . . . .	620
6.89	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	621
6.89.1	Detailed Description . . . . .	621
6.89.2	Constructor & Destructor Documentation . . . . .	622
6.89.2.1	ActiveMQTempQueueMarshaller . . . . .	622

6.89.2.2	~ActiveMQTempQueueMarshaller . . . . .	622
6.89.3	Member Function Documentation . . . . .	622
6.89.3.1	createObject . . . . .	622
6.89.3.2	getDataStructureType . . . . .	622
6.89.3.3	looseMarshal . . . . .	622
6.89.3.4	looseUnmarshal . . . . .	623
6.89.3.5	tightMarshal1 . . . . .	623
6.89.3.6	tightMarshal2 . . . . .	623
6.89.3.7	tightUnmarshal . . . . .	624
6.90	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference . . . . .	625
6.90.1	Detailed Description . . . . .	625
6.90.2	Constructor & Destructor Documentation . . . . .	626
6.90.2.1	ActiveMQTempQueueMarshaller . . . . .	626
6.90.2.2	~ActiveMQTempQueueMarshaller . . . . .	626
6.90.3	Member Function Documentation . . . . .	626
6.90.3.1	createObject . . . . .	626
6.90.3.2	getDataStructureType . . . . .	626
6.90.3.3	looseMarshal . . . . .	626
6.90.3.4	looseUnmarshal . . . . .	627
6.90.3.5	tightMarshal1 . . . . .	627
6.90.3.6	tightMarshal2 . . . . .	627
6.90.3.7	tightUnmarshal . . . . .	628
6.91	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller Class Reference . . . . .	629
6.91.1	Detailed Description . . . . .	629
6.91.2	Constructor & Destructor Documentation . . . . .	630
6.91.2.1	ActiveMQTempQueueMarshaller . . . . .	630
6.91.2.2	~ActiveMQTempQueueMarshaller . . . . .	630
6.91.3	Member Function Documentation . . . . .	630
6.91.3.1	createObject . . . . .	630
6.91.3.2	getDataStructureType . . . . .	630
6.91.3.3	looseMarshal . . . . .	630
6.91.3.4	looseUnmarshal . . . . .	631
6.91.3.5	tightMarshal1 . . . . .	631
6.91.3.6	tightMarshal2 . . . . .	631
6.91.3.7	tightUnmarshal . . . . .	632

6.92	activemq::commands::ActiveMQTempTopic Class Reference . . . . .	633
6.92.1	Constructor & Destructor Documentation . . . . .	634
6.92.1.1	ActiveMQTempTopic . . . . .	634
6.92.1.2	ActiveMQTempTopic . . . . .	634
6.92.1.3	~ActiveMQTempTopic . . . . .	634
6.92.2	Member Function Documentation . . . . .	634
6.92.2.1	clone . . . . .	634
6.92.2.2	cloneDataStructure . . . . .	634
6.92.2.3	copy . . . . .	634
6.92.2.4	copyDataStructure . . . . .	635
6.92.2.5	destroy . . . . .	635
6.92.2.6	equals . . . . .	635
6.92.2.7	getCMSDestination . . . . .	635
6.92.2.8	getCMSProperties . . . . .	635
6.92.2.9	getDataStructureType . . . . .	636
6.92.2.10	getDestinationType . . . . .	636
6.92.2.11	getTopicName . . . . .	636
6.92.2.12	toString . . . . .	636
6.92.3	Field Documentation . . . . .	637
6.92.3.1	ID_ACTIVEMQTEMPTOPIC . . . . .	637
6.93	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller Class Reference . . . . .	638
6.93.1	Detailed Description . . . . .	638
6.93.2	Constructor & Destructor Documentation . . . . .	639
6.93.2.1	ActiveMQTempTopicMarshaller . . . . .	639
6.93.2.2	~ActiveMQTempTopicMarshaller . . . . .	639
6.93.3	Member Function Documentation . . . . .	639
6.93.3.1	createObject . . . . .	639
6.93.3.2	getDataStructureType . . . . .	639
6.93.3.3	looseMarshal . . . . .	639
6.93.3.4	looseUnmarshal . . . . .	640
6.93.3.5	tightMarshal1 . . . . .	640
6.93.3.6	tightMarshal2 . . . . .	640
6.93.3.7	tightUnmarshal . . . . .	641
6.94	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller Class Reference . . . . .	642
6.94.1	Detailed Description . . . . .	642

6.94.2	Constructor & Destructor Documentation . . . . .	643
6.94.2.1	ActiveMQTempTopicMarshaller . . . . .	643
6.94.2.2	~ActiveMQTempTopicMarshaller . . . . .	643
6.94.3	Member Function Documentation . . . . .	643
6.94.3.1	createObject . . . . .	643
6.94.3.2	getDataStructureType . . . . .	643
6.94.3.3	looseMarshal . . . . .	643
6.94.3.4	looseUnmarshal . . . . .	644
6.94.3.5	tightMarshal1 . . . . .	644
6.94.3.6	tightMarshal2 . . . . .	644
6.94.3.7	tightUnmarshal . . . . .	645
6.95	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	646
6.95.1	Detailed Description . . . . .	646
6.95.2	Constructor & Destructor Documentation . . . . .	647
6.95.2.1	ActiveMQTempTopicMarshaller . . . . .	647
6.95.2.2	~ActiveMQTempTopicMarshaller . . . . .	647
6.95.3	Member Function Documentation . . . . .	647
6.95.3.1	createObject . . . . .	647
6.95.3.2	getDataStructureType . . . . .	647
6.95.3.3	looseMarshal . . . . .	647
6.95.3.4	looseUnmarshal . . . . .	648
6.95.3.5	tightMarshal1 . . . . .	648
6.95.3.6	tightMarshal2 . . . . .	648
6.95.3.7	tightUnmarshal . . . . .	649
6.96	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	650
6.96.1	Detailed Description . . . . .	650
6.96.2	Constructor & Destructor Documentation . . . . .	651
6.96.2.1	ActiveMQTempTopicMarshaller . . . . .	651
6.96.2.2	~ActiveMQTempTopicMarshaller . . . . .	651
6.96.3	Member Function Documentation . . . . .	651
6.96.3.1	createObject . . . . .	651
6.96.3.2	getDataStructureType . . . . .	651
6.96.3.3	looseMarshal . . . . .	651
6.96.3.4	looseUnmarshal . . . . .	652
6.96.3.5	tightMarshal1 . . . . .	652

6.96.3.6	tightMarshal2	652
6.96.3.7	tightUnmarshal	653
6.97	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	
	Class Reference	654
6.97.1	Detailed Description	654
6.97.2	Constructor & Destructor Documentation	655
6.97.2.1	ActiveMQTempTopicMarshaller	655
6.97.2.2	~ActiveMQTempTopicMarshaller	655
6.97.3	Member Function Documentation	655
6.97.3.1	createObject	655
6.97.3.2	getDataStructureType	655
6.97.3.3	looseMarshal	655
6.97.3.4	looseUnmarshal	656
6.97.3.5	tightMarshal1	656
6.97.3.6	tightMarshal2	656
6.97.3.7	tightUnmarshal	657
6.98	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	
	Class Reference	658
6.98.1	Detailed Description	658
6.98.2	Constructor & Destructor Documentation	659
6.98.2.1	ActiveMQTempTopicMarshaller	659
6.98.2.2	~ActiveMQTempTopicMarshaller	659
6.98.3	Member Function Documentation	659
6.98.3.1	createObject	659
6.98.3.2	getDataStructureType	659
6.98.3.3	looseMarshal	659
6.98.3.4	looseUnmarshal	660
6.98.3.5	tightMarshal1	660
6.98.3.6	tightMarshal2	660
6.98.3.7	tightUnmarshal	661
6.99	activemq::commands::ActiveMQTextMessage Class Reference	662
6.99.1	Constructor & Destructor Documentation	663
6.99.1.1	ActiveMQTextMessage	663
6.99.1.2	~ActiveMQTextMessage	663
6.99.2	Member Function Documentation	663
6.99.2.1	beforeMarshal	663
6.99.2.2	clearBody	663

6.99.2.3	clone . . . . .	663
6.99.2.4	cloneDataStructure . . . . .	664
6.99.2.5	copyDataStructure . . . . .	664
6.99.2.6	equals . . . . .	664
6.99.2.7	getDataStructureType . . . . .	664
6.99.2.8	getSize . . . . .	665
6.99.2.9	getText . . . . .	665
6.99.2.10	setText . . . . .	665
6.99.2.11	setText . . . . .	665
6.99.2.12	toString . . . . .	666
6.99.3	Field Documentation . . . . .	666
6.99.3.1	ID_ACTIVEMQTEXTMESSAGE . . . . .	666
6.99.3.2	text . . . . .	666
6.100	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	
	Class Reference . . . . .	667
6.100.1	Detailed Description . . . . .	667
6.100.2	Constructor & Destructor Documentation . . . . .	668
6.100.2.1	ActiveMQTextMessageMarshaller . . . . .	668
6.100.2.2	~ActiveMQTextMessageMarshaller . . . . .	668
6.100.3	Member Function Documentation . . . . .	668
6.100.3.1	createObject . . . . .	668
6.100.3.2	getDataStructureType . . . . .	668
6.100.3.3	looseMarshal . . . . .	668
6.100.3.4	looseUnmarshal . . . . .	669
6.100.3.5	tightMarshal1 . . . . .	669
6.100.3.6	tightMarshal2 . . . . .	669
6.100.3.7	tightUnmarshal . . . . .	670
6.101	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	
	Class Reference . . . . .	671
6.101.1	Detailed Description . . . . .	671
6.101.2	Constructor & Destructor Documentation . . . . .	672
6.101.2.1	ActiveMQTextMessageMarshaller . . . . .	672
6.101.2.2	~ActiveMQTextMessageMarshaller . . . . .	672
6.101.3	Member Function Documentation . . . . .	672
6.101.3.1	createObject . . . . .	672
6.101.3.2	getDataStructureType . . . . .	672
6.101.3.3	looseMarshal . . . . .	672

6.101.3.4 looseUnmarshal . . . . .	673
6.101.3.5 tightMarshal1 . . . . .	673
6.101.3.6 tightMarshal2 . . . . .	673
6.101.3.7 tightUnmarshal . . . . .	674
6.102activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	675
6.102.1 Detailed Description . . . . .	675
6.102.2 Constructor & Destructor Documentation . . . . .	676
6.102.2.1 ActiveMQTextMessageMarshaller . . . . .	676
6.102.2.2 ~ActiveMQTextMessageMarshaller . . . . .	676
6.102.3 Member Function Documentation . . . . .	676
6.102.3.1 createObject . . . . .	676
6.102.3.2 getDataStructureType . . . . .	676
6.102.3.3 looseMarshal . . . . .	676
6.102.3.4 looseUnmarshal . . . . .	677
6.102.3.5 tightMarshal1 . . . . .	677
6.102.3.6 tightMarshal2 . . . . .	677
6.102.3.7 tightUnmarshal . . . . .	678
6.103activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	679
6.103.1 Detailed Description . . . . .	679
6.103.2 Constructor & Destructor Documentation . . . . .	680
6.103.2.1 ActiveMQTextMessageMarshaller . . . . .	680
6.103.2.2 ~ActiveMQTextMessageMarshaller . . . . .	680
6.103.3 Member Function Documentation . . . . .	680
6.103.3.1 createObject . . . . .	680
6.103.3.2 getDataStructureType . . . . .	680
6.103.3.3 looseMarshal . . . . .	680
6.103.3.4 looseUnmarshal . . . . .	681
6.103.3.5 tightMarshal1 . . . . .	681
6.103.3.6 tightMarshal2 . . . . .	681
6.103.3.7 tightUnmarshal . . . . .	682
6.104activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	683
6.104.1 Detailed Description . . . . .	683
6.104.2 Constructor & Destructor Documentation . . . . .	684
6.104.2.1 ActiveMQTextMessageMarshaller . . . . .	684



6.104.2.2 ~ActiveMQTextMessageMarshaller . . . . .	684
6.104.3 Member Function Documentation . . . . .	684
6.104.3.1 createObject . . . . .	684
6.104.3.2 getDataStructureType . . . . .	684
6.104.3.3 looseMarshal . . . . .	684
6.104.3.4 looseUnmarshal . . . . .	685
6.104.3.5 tightMarshal1 . . . . .	685
6.104.3.6 tightMarshal2 . . . . .	685
6.104.3.7 tightUnmarshal . . . . .	686
6.105activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller Class Reference . . . . .	687
6.105.1 Detailed Description . . . . .	687
6.105.2 Constructor & Destructor Documentation . . . . .	688
6.105.2.1 ActiveMQTextMessageMarshaller . . . . .	688
6.105.2.2 ~ActiveMQTextMessageMarshaller . . . . .	688
6.105.3 Member Function Documentation . . . . .	688
6.105.3.1 createObject . . . . .	688
6.105.3.2 getDataStructureType . . . . .	688
6.105.3.3 looseMarshal . . . . .	688
6.105.3.4 looseUnmarshal . . . . .	689
6.105.3.5 tightMarshal1 . . . . .	689
6.105.3.6 tightMarshal2 . . . . .	689
6.105.3.7 tightUnmarshal . . . . .	690
6.106activemq::commands::ActiveMQTopic Class Reference . . . . .	691
6.106.1 Constructor & Destructor Documentation . . . . .	692
6.106.1.1 ActiveMQTopic . . . . .	692
6.106.1.2 ActiveMQTopic . . . . .	692
6.106.1.3 ~ActiveMQTopic . . . . .	692
6.106.2 Member Function Documentation . . . . .	692
6.106.2.1 clone . . . . .	692
6.106.2.2 cloneDataStructure . . . . .	692
6.106.2.3 copy . . . . .	692
6.106.2.4 copyDataStructure . . . . .	692
6.106.2.5 equals . . . . .	693
6.106.2.6 getCMSDestination . . . . .	693
6.106.2.7 getCMSProperties . . . . .	693
6.106.2.8 getDataStructureType . . . . .	693

6.106.2.9	getDestinationType . . . . .	693
6.106.2.10	getTopicName . . . . .	694
6.106.2.11	toString . . . . .	694
6.106.3	Field Documentation . . . . .	694
6.106.3.1	ID_ACTIVEMQTOPIC . . . . .	694
6.107	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference . . . . .	695
6.107.1	Detailed Description . . . . .	695
6.107.2	Constructor & Destructor Documentation . . . . .	696
6.107.2.1	ActiveMQTopicMarshaller . . . . .	696
6.107.2.2	~ActiveMQTopicMarshaller . . . . .	696
6.107.3	Member Function Documentation . . . . .	696
6.107.3.1	createObject . . . . .	696
6.107.3.2	getDataStructureType . . . . .	696
6.107.3.3	looseMarshal . . . . .	696
6.107.3.4	looseUnmarshal . . . . .	697
6.107.3.5	tightMarshal1 . . . . .	697
6.107.3.6	tightMarshal2 . . . . .	697
6.107.3.7	tightUnmarshal . . . . .	698
6.108	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference . . . . .	699
6.108.1	Detailed Description . . . . .	699
6.108.2	Constructor & Destructor Documentation . . . . .	700
6.108.2.1	ActiveMQTopicMarshaller . . . . .	700
6.108.2.2	~ActiveMQTopicMarshaller . . . . .	700
6.108.3	Member Function Documentation . . . . .	700
6.108.3.1	createObject . . . . .	700
6.108.3.2	getDataStructureType . . . . .	700
6.108.3.3	looseMarshal . . . . .	700
6.108.3.4	looseUnmarshal . . . . .	701
6.108.3.5	tightMarshal1 . . . . .	701
6.108.3.6	tightMarshal2 . . . . .	701
6.108.3.7	tightUnmarshal . . . . .	702
6.109	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference . . . . .	703
6.109.1	Detailed Description . . . . .	703
6.109.2	Constructor & Destructor Documentation . . . . .	704

6.109.2.1 ActiveMQTopicMarshaller . . . . .	704
6.109.2.2 ~ActiveMQTopicMarshaller . . . . .	704
6.109.3 Member Function Documentation . . . . .	704
6.109.3.1 createObject . . . . .	704
6.109.3.2 getDataStructureType . . . . .	704
6.109.3.3 looseMarshal . . . . .	704
6.109.3.4 looseUnmarshal . . . . .	705
6.109.3.5 tightMarshal1 . . . . .	705
6.109.3.6 tightMarshal2 . . . . .	705
6.109.3.7 tightUnmarshal . . . . .	706
6.110activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference . . . . .	707
6.110.1 Detailed Description . . . . .	707
6.110.2 Constructor & Destructor Documentation . . . . .	708
6.110.2.1 ActiveMQTopicMarshaller . . . . .	708
6.110.2.2 ~ActiveMQTopicMarshaller . . . . .	708
6.110.3 Member Function Documentation . . . . .	708
6.110.3.1 createObject . . . . .	708
6.110.3.2 getDataStructureType . . . . .	708
6.110.3.3 looseMarshal . . . . .	708
6.110.3.4 looseUnmarshal . . . . .	709
6.110.3.5 tightMarshal1 . . . . .	709
6.110.3.6 tightMarshal2 . . . . .	709
6.110.3.7 tightUnmarshal . . . . .	710
6.111activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller Class Reference . . . . .	711
6.111.1 Detailed Description . . . . .	711
6.111.2 Constructor & Destructor Documentation . . . . .	712
6.111.2.1 ActiveMQTopicMarshaller . . . . .	712
6.111.2.2 ~ActiveMQTopicMarshaller . . . . .	712
6.111.3 Member Function Documentation . . . . .	712
6.111.3.1 createObject . . . . .	712
6.111.3.2 getDataStructureType . . . . .	712
6.111.3.3 looseMarshal . . . . .	712
6.111.3.4 looseUnmarshal . . . . .	713
6.111.3.5 tightMarshal1 . . . . .	713
6.111.3.6 tightMarshal2 . . . . .	713

6.111.3.7	tightUnmarshal . . . . .	714
6.112	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference . . . . .	715
6.112.1	Detailed Description . . . . .	715
6.112.2	Constructor & Destructor Documentation . . . . .	716
6.112.2.1	ActiveMQTopicMarshaller . . . . .	716
6.112.2.2	~ActiveMQTopicMarshaller . . . . .	716
6.112.3	Member Function Documentation . . . . .	716
6.112.3.1	createObject . . . . .	716
6.112.3.2	getDataStructureType . . . . .	716
6.112.3.3	looseMarshal . . . . .	716
6.112.3.4	looseUnmarshal . . . . .	717
6.112.3.5	tightMarshal1 . . . . .	717
6.112.3.6	tightMarshal2 . . . . .	717
6.112.3.7	tightUnmarshal . . . . .	718
6.113	activemq::core::ActiveMQTransactionContext Class Reference . . . . .	719
6.113.1	Detailed Description . . . . .	719
6.113.2	Constructor & Destructor Documentation . . . . .	720
6.113.2.1	ActiveMQTransactionContext . . . . .	720
6.113.2.2	~ActiveMQTransactionContext . . . . .	720
6.113.3	Member Function Documentation . . . . .	720
6.113.3.1	addSynchronization . . . . .	720
6.113.3.2	begin . . . . .	720
6.113.3.3	commit . . . . .	720
6.113.3.4	getTransactionId . . . . .	720
6.113.3.5	isInTransaction . . . . .	721
6.113.3.6	removeSynchronization . . . . .	721
6.113.3.7	rollback . . . . .	721
6.114	decaf::util::zip::Adler32 Class Reference . . . . .	722
6.114.1	Detailed Description . . . . .	722
6.114.2	Constructor & Destructor Documentation . . . . .	723
6.114.2.1	Adler32 . . . . .	723
6.114.2.2	~Adler32 . . . . .	723
6.114.3	Member Function Documentation . . . . .	723
6.114.3.1	getValue . . . . .	723
6.114.3.2	reset . . . . .	723
6.114.3.3	update . . . . .	723

6.114.3.4 update . . . . .	723
6.114.3.5 update . . . . .	724
6.114.3.6 update . . . . .	724
6.115decaf::lang::Appendable Class Reference . . . . .	725
6.115.1 Detailed Description . . . . .	725
6.115.2 Constructor & Destructor Documentation . . . . .	726
6.115.2.1 ~Appendable . . . . .	726
6.115.3 Member Function Documentation . . . . .	726
6.115.3.1 append . . . . .	726
6.115.3.2 append . . . . .	726
6.115.3.3 append . . . . .	726
6.116decaf::internal::AprPool Class Reference . . . . .	728
6.116.1 Detailed Description . . . . .	728
6.116.2 Constructor & Destructor Documentation . . . . .	728
6.116.2.1 AprPool . . . . .	728
6.116.2.2 ~AprPool . . . . .	728
6.116.3 Member Function Documentation . . . . .	728
6.116.3.1 cleanup . . . . .	728
6.116.3.2 getAprPool . . . . .	728
6.116.3.3 getGlobalPool . . . . .	729
6.117decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference . . . . .	730
6.117.1 Detailed Description . . . . .	731
6.117.2 Member Typedef Documentation . . . . .	732
6.117.2.1 ConstReferenceType . . . . .	732
6.117.2.2 CounterType . . . . .	732
6.117.2.3 PointerType . . . . .	732
6.117.2.4 ReferenceType . . . . .	732
6.117.3 Constructor & Destructor Documentation . . . . .	732
6.117.3.1 ArrayPointer . . . . .	732
6.117.3.2 ArrayPointer . . . . .	732
6.117.3.3 ArrayPointer . . . . .	732
6.117.3.4 ArrayPointer . . . . .	733
6.117.3.5 ~ArrayPointer . . . . .	733
6.117.4 Member Function Documentation . . . . .	733
6.117.4.1 clone . . . . .	733
6.117.4.2 get . . . . .	733

6.117.4.3 length . . . . .	734
6.117.4.4 operator! . . . . .	734
6.117.4.5 operator!= . . . . .	734
6.117.4.6 operator= . . . . .	734
6.117.4.7 operator= . . . . .	734
6.117.4.8 operator== . . . . .	735
6.117.4.9 operator[] . . . . .	735
6.117.4.10operator[] . . . . .	735
6.117.4.11release . . . . .	735
6.117.4.12reset . . . . .	735
6.117.4.13wap . . . . .	736
6.117.5 Friends And Related Function Documentation . . . . .	736
6.117.5.1 operator!= . . . . .	736
6.117.5.2 operator!= . . . . .	736
6.117.5.3 operator== . . . . .	736
6.117.5.4 operator== . . . . .	736
6.118decaf::lang::ArrayPointerComparator< T, R > Class Template Reference . . . . .	737
6.118.1 Detailed Description . . . . .	737
6.118.2 Member Function Documentation . . . . .	737
6.118.2.1 compare . . . . .	737
6.118.2.2 operator() . . . . .	737
6.119decaf::util::concurrent::atomic::AtomicBoolean Class Reference . . . . .	738
6.119.1 Detailed Description . . . . .	738
6.119.2 Constructor & Destructor Documentation . . . . .	738
6.119.2.1 AtomicBoolean . . . . .	738
6.119.2.2 AtomicBoolean . . . . .	738
6.119.2.3 ~AtomicBoolean . . . . .	739
6.119.3 Member Function Documentation . . . . .	739
6.119.3.1 compareAndSet . . . . .	739
6.119.3.2 get . . . . .	739
6.119.3.3 getAndSet . . . . .	739
6.119.3.4 set . . . . .	739
6.119.3.5 toString . . . . .	740
6.120decaf::util::concurrent::atomic::AtomicInteger Class Reference . . . . .	741
6.120.1 Detailed Description . . . . .	742
6.120.2 Constructor & Destructor Documentation . . . . .	742

6.120.2.1 AtomicInteger . . . . .	742
6.120.2.2 AtomicInteger . . . . .	742
6.120.2.3 ~AtomicInteger . . . . .	742
6.120.3 Member Function Documentation . . . . .	742
6.120.3.1 addAndGet . . . . .	742
6.120.3.2 compareAndSet . . . . .	743
6.120.3.3 decrementAndGet . . . . .	743
6.120.3.4 doubleValue . . . . .	743
6.120.3.5 floatValue . . . . .	743
6.120.3.6 get . . . . .	744
6.120.3.7 getAndAdd . . . . .	744
6.120.3.8 getAndDecrement . . . . .	744
6.120.3.9 getAndIncrement . . . . .	744
6.120.3.10 getAndSet . . . . .	744
6.120.3.11 incrementAndGet . . . . .	745
6.120.3.12 intValue . . . . .	745
6.120.3.13 longValue . . . . .	745
6.120.3.14 set . . . . .	745
6.120.3.15 toString . . . . .	745
6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference . . . . .	746
6.121.1 Constructor & Destructor Documentation . . . . .	747
6.121.1.1 AtomicRefCounter . . . . .	747
6.121.1.2 AtomicRefCounter . . . . .	747
6.121.1.3 ~AtomicRefCounter . . . . .	747
6.121.2 Member Function Documentation . . . . .	747
6.121.2.1 release . . . . .	747
6.121.2.2 swap . . . . .	748
6.122 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference . . . . .	749
6.122.1 Detailed Description . . . . .	749
6.122.2 Constructor & Destructor Documentation . . . . .	750
6.122.2.1 AtomicReference . . . . .	750
6.122.2.2 AtomicReference . . . . .	750
6.122.2.3 ~AtomicReference . . . . .	750
6.122.3 Member Function Documentation . . . . .	750
6.122.3.1 compareAndSet . . . . .	750
6.122.3.2 get . . . . .	750

6.122.3.3	getAndSet . . . . .	750
6.122.3.4	set . . . . .	751
6.122.3.5	toString . . . . .	751
6.123	activemq::transport::failover::BackupTransport Class Reference . . . . .	752
6.123.1	Constructor & Destructor Documentation . . . . .	752
6.123.1.1	BackupTransport . . . . .	752
6.123.1.2	~BackupTransport . . . . .	752
6.123.2	Member Function Documentation . . . . .	752
6.123.2.1	getTransport . . . . .	752
6.123.2.2	getUri . . . . .	753
6.123.2.3	isClosed . . . . .	753
6.123.2.4	onException . . . . .	753
6.123.2.5	setClosed . . . . .	753
6.123.2.6	setTransport . . . . .	753
6.123.2.7	setUri . . . . .	754
6.124	activemq::transport::failover::BackupTransportPool Class Reference . . . . .	755
6.124.1	Constructor & Destructor Documentation . . . . .	756
6.124.1.1	BackupTransportPool . . . . .	756
6.124.1.2	BackupTransportPool . . . . .	756
6.124.1.3	~BackupTransportPool . . . . .	756
6.124.2	Member Function Documentation . . . . .	756
6.124.2.1	getBackup . . . . .	756
6.124.2.2	getBackupPoolSize . . . . .	756
6.124.2.3	isEnabled . . . . .	756
6.124.2.4	isPending . . . . .	757
6.124.2.5	iterate . . . . .	757
6.124.2.6	setBackupPoolSize . . . . .	757
6.124.2.7	setEnabled . . . . .	757
6.124.3	Friends And Related Function Documentation . . . . .	757
6.124.3.1	BackupTransport . . . . .	757
6.125	activemq::commands::BaseCommand Class Reference . . . . .	758
6.125.1	Constructor & Destructor Documentation . . . . .	759
6.125.1.1	BaseCommand . . . . .	759
6.125.1.2	~BaseCommand . . . . .	759
6.125.2	Member Function Documentation . . . . .	759
6.125.2.1	copyDataStructure . . . . .	759



6.125.2.2 equals . . . . .	759
6.125.2.3 getCommandId . . . . .	760
6.125.2.4 isBrokerInfo . . . . .	760
6.125.2.5 isConnectionInfo . . . . .	761
6.125.2.6 isConsumerInfo . . . . .	761
6.125.2.7 isKeepAliveInfo . . . . .	761
6.125.2.8 isMessage . . . . .	761
6.125.2.9 isMessageAck . . . . .	761
6.125.2.10sMessageDispatch . . . . .	761
6.125.2.11sMessageDispatchNotification . . . . .	761
6.125.2.12sProducerAck . . . . .	762
6.125.2.13sProducerInfo . . . . .	762
6.125.2.14sRemoveInfo . . . . .	762
6.125.2.15sRemoveSubscriptionInfo . . . . .	762
6.125.2.16sResponse . . . . .	762
6.125.2.17sResponseRequired . . . . .	762
6.125.2.18sShutdownInfo . . . . .	762
6.125.2.19sTransactionInfo . . . . .	763
6.125.2.20sWireFormatInfo . . . . .	763
6.125.2.21setCommandId . . . . .	763
6.125.2.22setResponseRequired . . . . .	763
6.125.2.23oString . . . . .	763
6.126activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference . . . . .	765
6.126.1 Detailed Description . . . . .	765
6.126.2 Constructor & Destructor Documentation . . . . .	766
6.126.2.1 BaseCommandMarshaller . . . . .	766
6.126.2.2 ~BaseCommandMarshaller . . . . .	766
6.126.3 Member Function Documentation . . . . .	766
6.126.3.1 looseMarshal . . . . .	766
6.126.3.2 looseUnmarshal . . . . .	767
6.126.3.3 tightMarshal1 . . . . .	768
6.126.3.4 tightMarshal2 . . . . .	769
6.126.3.5 tightUnmarshal . . . . .	770
6.127activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference . . . . .	772
6.127.1 Detailed Description . . . . .	772

6.127.2 Constructor & Destructor Documentation . . . . .	773
6.127.2.1 BaseCommandMarshaller . . . . .	773
6.127.2.2 ~BaseCommandMarshaller . . . . .	773
6.127.3 Member Function Documentation . . . . .	773
6.127.3.1 looseMarshal . . . . .	773
6.127.3.2 looseUnmarshal . . . . .	774
6.127.3.3 tightMarshal1 . . . . .	775
6.127.3.4 tightMarshal2 . . . . .	776
6.127.3.5 tightUnmarshal . . . . .	777
6.128activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference . . . . .	779
6.128.1 Detailed Description . . . . .	779
6.128.2 Constructor & Destructor Documentation . . . . .	780
6.128.2.1 BaseCommandMarshaller . . . . .	780
6.128.2.2 ~BaseCommandMarshaller . . . . .	780
6.128.3 Member Function Documentation . . . . .	780
6.128.3.1 looseMarshal . . . . .	780
6.128.3.2 looseUnmarshal . . . . .	781
6.128.3.3 tightMarshal1 . . . . .	782
6.128.3.4 tightMarshal2 . . . . .	783
6.128.3.5 tightUnmarshal . . . . .	784
6.129activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference . . . . .	786
6.129.1 Detailed Description . . . . .	786
6.129.2 Constructor & Destructor Documentation . . . . .	787
6.129.2.1 BaseCommandMarshaller . . . . .	787
6.129.2.2 ~BaseCommandMarshaller . . . . .	787
6.129.3 Member Function Documentation . . . . .	787
6.129.3.1 looseMarshal . . . . .	787
6.129.3.2 looseUnmarshal . . . . .	788
6.129.3.3 tightMarshal1 . . . . .	789
6.129.3.4 tightMarshal2 . . . . .	790
6.129.3.5 tightUnmarshal . . . . .	791
6.130activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller Class Reference . . . . .	793
6.130.1 Detailed Description . . . . .	793
6.130.2 Constructor & Destructor Documentation . . . . .	794

6.130.2.1 BaseCommandMarshaller . . . . .	794
6.130.2.2 ~BaseCommandMarshaller . . . . .	794
6.130.3 Member Function Documentation . . . . .	794
6.130.3.1 looseMarshal . . . . .	794
6.130.3.2 looseUnmarshal . . . . .	795
6.130.3.3 tightMarshal1 . . . . .	796
6.130.3.4 tightMarshal2 . . . . .	797
6.130.3.5 tightUnmarshal . . . . .	798
6.131activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference . . . . .	800
6.131.1 Detailed Description . . . . .	800
6.131.2 Constructor & Destructor Documentation . . . . .	801
6.131.2.1 BaseCommandMarshaller . . . . .	801
6.131.2.2 ~BaseCommandMarshaller . . . . .	801
6.131.3 Member Function Documentation . . . . .	801
6.131.3.1 looseMarshal . . . . .	801
6.131.3.2 looseUnmarshal . . . . .	802
6.131.3.3 tightMarshal1 . . . . .	803
6.131.3.4 tightMarshal2 . . . . .	804
6.131.3.5 tightUnmarshal . . . . .	805
6.132activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference . . . . .	807
6.132.1 Detailed Description . . . . .	813
6.132.2 Constructor & Destructor Documentation . . . . .	813
6.132.2.1 ~BaseDataStreamMarshaller . . . . .	813
6.132.3 Member Function Documentation . . . . .	813
6.132.3.1 looseMarshal . . . . .	813
6.132.3.2 looseMarshalBrokerError . . . . .	814
6.132.3.3 looseMarshalCachedObject . . . . .	814
6.132.3.4 looseMarshalLong . . . . .	814
6.132.3.5 looseMarshalNestedObject . . . . .	815
6.132.3.6 looseMarshalObjectArray . . . . .	815
6.132.3.7 looseMarshalString . . . . .	816
6.132.3.8 looseUnmarshal . . . . .	816
6.132.3.9 looseUnmarshalBrokerError . . . . .	816
6.132.3.10looseUnmarshalByteArray . . . . .	817
6.132.3.11looseUnmarshalCachedObject . . . . .	817

6.132.3.12	ooseUnmarshalConstByteArray . . . . .	817
6.132.3.13	ooseUnmarshalLong . . . . .	818
6.132.3.14	ooseUnmarshalNestedObject . . . . .	818
6.132.3.15	ooseUnmarshalString . . . . .	818
6.132.3.16	readAsciiString . . . . .	819
6.132.3.17	ightMarshal1 . . . . .	819
6.132.3.18	ightMarshal2 . . . . .	819
6.132.3.19	ightMarshalBrokerError1 . . . . .	820
6.132.3.20	ightMarshalBrokerError2 . . . . .	820
6.132.3.21	ightMarshalCachedObject1 . . . . .	821
6.132.3.22	ightMarshalCachedObject2 . . . . .	821
6.132.3.23	ightMarshalLong1 . . . . .	821
6.132.3.24	ightMarshalLong2 . . . . .	822
6.132.3.25	ightMarshalNestedObject1 . . . . .	822
6.132.3.26	ightMarshalNestedObject2 . . . . .	823
6.132.3.27	ightMarshalObjectArray1 . . . . .	823
6.132.3.28	ightMarshalObjectArray2 . . . . .	823
6.132.3.29	ightMarshalString1 . . . . .	824
6.132.3.30	ightMarshalString2 . . . . .	824
6.132.3.31	ightUnmarshal . . . . .	825
6.132.3.32	ightUnmarshalBrokerError . . . . .	825
6.132.3.33	ightUnmarshalByteArray . . . . .	825
6.132.3.34	ightUnmarshalCachedObject . . . . .	826
6.132.3.35	ightUnmarshalConstByteArray . . . . .	826
6.132.3.36	ightUnmarshalLong . . . . .	827
6.132.3.37	ightUnmarshalNestedObject . . . . .	827
6.132.3.38	ightUnmarshalString . . . . .	827
6.132.3.39	oHexFromBytes . . . . .	828
6.132.3.40	oString . . . . .	828
6.132.3.41	toString . . . . .	828
6.132.3.42	oString . . . . .	829
6.133	activemq::commands::BaseDataStructure Class Reference . . . . .	830
6.133.1	Constructor & Destructor Documentation . . . . .	831
6.133.1.1	~BaseDataStructure . . . . .	831
6.133.2	Member Function Documentation . . . . .	831
6.133.2.1	afterMarshal . . . . .	831

6.133.2.2 afterUnmarshal . . . . .	831
6.133.2.3 beforeMarshal . . . . .	831
6.133.2.4 beforeUnmarshal . . . . .	831
6.133.2.5 copyDataStructure . . . . .	832
6.133.2.6 equals . . . . .	832
6.133.2.7 getMarshaledForm . . . . .	832
6.133.2.8 isMarshalAware . . . . .	832
6.133.2.9 setMarshaledForm . . . . .	833
6.133.2.10 toString . . . . .	833
6.134binary_function Class Reference . . . . .	835
6.135decaf::net::BindException Class Reference . . . . .	836
6.135.1 Constructor & Destructor Documentation . . . . .	836
6.135.1.1 BindException . . . . .	836
6.135.1.2 BindException . . . . .	836
6.135.1.3 BindException . . . . .	837
6.135.1.4 BindException . . . . .	837
6.135.1.5 BindException . . . . .	837
6.135.1.6 BindException . . . . .	837
6.135.1.7 ~BindException . . . . .	838
6.135.2 Member Function Documentation . . . . .	838
6.135.2.1 clone . . . . .	838
6.136decaf::io::BlockingByteArrayInputStream Class Reference . . . . .	839
6.136.1 Detailed Description . . . . .	840
6.136.2 Constructor & Destructor Documentation . . . . .	840
6.136.2.1 BlockingByteArrayInputStream . . . . .	840
6.136.2.2 BlockingByteArrayInputStream . . . . .	840
6.136.2.3 ~BlockingByteArrayInputStream . . . . .	840
6.136.3 Member Function Documentation . . . . .	840
6.136.3.1 available . . . . .	840
6.136.3.2 close . . . . .	841
6.136.3.3 doReadArrayBounded . . . . .	841
6.136.3.4 doReadByte . . . . .	841
6.136.3.5 setByteArray . . . . .	841
6.136.3.6 skip . . . . .	841
6.137decaf::util::concurrent::BlockingQueue< E > Class Template Reference . . . . .	843
6.137.1 Detailed Description . . . . .	844

6.137.2 Constructor & Destructor Documentation . . . . .	846
6.137.2.1 ~BlockingQueue . . . . .	846
6.137.3 Member Function Documentation . . . . .	846
6.137.3.1 drainTo . . . . .	846
6.137.3.2 drainTo . . . . .	846
6.137.3.3 offer . . . . .	847
6.137.3.4 poll . . . . .	847
6.137.3.5 put . . . . .	848
6.137.3.6 remainingCapacity . . . . .	848
6.137.3.7 take . . . . .	849
6.138decaf::lang::Boolean Class Reference . . . . .	850
6.138.1 Constructor & Destructor Documentation . . . . .	851
6.138.1.1 Boolean . . . . .	851
6.138.1.2 Boolean . . . . .	851
6.138.1.3 ~Boolean . . . . .	851
6.138.2 Member Function Documentation . . . . .	851
6.138.2.1 booleanValue . . . . .	851
6.138.2.2 compareTo . . . . .	851
6.138.2.3 compareTo . . . . .	852
6.138.2.4 equals . . . . .	852
6.138.2.5 equals . . . . .	852
6.138.2.6 operator< . . . . .	852
6.138.2.7 operator< . . . . .	852
6.138.2.8 operator== . . . . .	853
6.138.2.9 operator== . . . . .	853
6.138.2.10parseBoolean . . . . .	853
6.138.2.11toString . . . . .	854
6.138.2.12toString . . . . .	854
6.138.2.13valueOf . . . . .	854
6.138.2.14valueOf . . . . .	854
6.138.3 Field Documentation . . . . .	854
6.138.3.1 _FALSE . . . . .	854
6.138.3.2 _TRUE . . . . .	854
6.139activemq::commands::BooleanExpression Class Reference . . . . .	855
6.139.1 Constructor & Destructor Documentation . . . . .	855
6.139.1.1 BooleanExpression . . . . .	855

6.139.1.2 <code>~BooleanExpression</code> . . . . .	855
6.139.2 Member Function Documentation . . . . .	855
6.139.2.1 <code>cloneDataStructure</code> . . . . .	855
6.139.2.2 <code>copyDataStructure</code> . . . . .	856
6.139.2.3 <code>equals</code> . . . . .	856
6.139.2.4 <code>toString</code> . . . . .	856
6.140 <code>activemq::wireformat::openwire::utils::BooleanStream</code> Class Reference . . . . .	857
6.140.1 Detailed Description . . . . .	857
6.140.2 Constructor & Destructor Documentation . . . . .	858
6.140.2.1 <code>BooleanStream</code> . . . . .	858
6.140.2.2 <code>~BooleanStream</code> . . . . .	858
6.140.3 Member Function Documentation . . . . .	858
6.140.3.1 <code>clear</code> . . . . .	858
6.140.3.2 <code>marshal</code> . . . . .	858
6.140.3.3 <code>marshal</code> . . . . .	858
6.140.3.4 <code>marshalledSize</code> . . . . .	858
6.140.3.5 <code>readBoolean</code> . . . . .	858
6.140.3.6 <code>unmarshal</code> . . . . .	859
6.140.3.7 <code>writeBoolean</code> . . . . .	859
6.141 <code>decaf::util::concurrent::BrokenBarrierException</code> Class Reference . . . . .	860
6.141.1 Constructor & Destructor Documentation . . . . .	860
6.141.1.1 <code>BrokenBarrierException</code> . . . . .	860
6.141.1.2 <code>BrokenBarrierException</code> . . . . .	860
6.141.1.3 <code>BrokenBarrierException</code> . . . . .	861
6.141.1.4 <code>BrokenBarrierException</code> . . . . .	861
6.141.1.5 <code>BrokenBarrierException</code> . . . . .	861
6.141.1.6 <code>BrokenBarrierException</code> . . . . .	861
6.141.1.7 <code>~BrokenBarrierException</code> . . . . .	862
6.141.2 Member Function Documentation . . . . .	862
6.141.2.1 <code>clone</code> . . . . .	862
6.142 <code>activemq::commands::BrokerError</code> Class Reference . . . . .	863
6.142.1 Detailed Description . . . . .	864
6.142.2 Constructor & Destructor Documentation . . . . .	864
6.142.2.1 <code>BrokerError</code> . . . . .	864
6.142.2.2 <code>~BrokerError</code> . . . . .	864
6.142.3 Member Function Documentation . . . . .	864

6.142.3.1 cloneDataStructure . . . . .	864
6.142.3.2 copyDataStructure . . . . .	864
6.142.3.3 getCause . . . . .	865
6.142.3.4 getDataStructureType . . . . .	865
6.142.3.5 getExceptionClass . . . . .	865
6.142.3.6 getMessage . . . . .	865
6.142.3.7 getStackTraceElements . . . . .	865
6.142.3.8 setCause . . . . .	866
6.142.3.9 setExceptionClass . . . . .	866
6.142.3.10 setMessage . . . . .	866
6.142.3.11 setStackTraceElements . . . . .	866
6.142.3.12 visit . . . . .	866
6.143activemq::exceptions::BrokerException Class Reference . . . . .	867
6.143.1 Constructor & Destructor Documentation . . . . .	867
6.143.1.1 BrokerException . . . . .	867
6.143.1.2 BrokerException . . . . .	867
6.143.1.3 BrokerException . . . . .	867
6.143.1.4 BrokerException . . . . .	867
6.143.1.5 BrokerException . . . . .	867
6.143.1.6 ~BrokerException . . . . .	867
6.143.2 Member Function Documentation . . . . .	867
6.143.2.1 clone . . . . .	867
6.144activemq::commands::BrokerId Class Reference . . . . .	869
6.144.1 Member Typedef Documentation . . . . .	870
6.144.1.1 COMPARATOR . . . . .	870
6.144.2 Constructor & Destructor Documentation . . . . .	870
6.144.2.1 BrokerId . . . . .	870
6.144.2.2 BrokerId . . . . .	870
6.144.2.3 ~BrokerId . . . . .	870
6.144.3 Member Function Documentation . . . . .	870
6.144.3.1 cloneDataStructure . . . . .	870
6.144.3.2 compareTo . . . . .	870
6.144.3.3 copyDataStructure . . . . .	870
6.144.3.4 equals . . . . .	870
6.144.3.5 equals . . . . .	870
6.144.3.6 getDataStructureType . . . . .	871



6.144.3.7	getValue	871
6.144.3.8	getValue	871
6.144.3.9	operator<	871
6.144.3.10	operator=	871
6.144.3.11	operator==	871
6.144.3.12	setValue	871
6.144.3.13	toString	871
6.144.4	Field Documentation	871
6.144.4.1	ID_BROKERID	871
6.144.4.2	value	871
6.145	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference	872
6.145.1	Detailed Description	872
6.145.2	Constructor & Destructor Documentation	873
6.145.2.1	BrokerIdMarshaller	873
6.145.2.2	~BrokerIdMarshaller	873
6.145.3	Member Function Documentation	873
6.145.3.1	createObject	873
6.145.3.2	getDataStructureType	873
6.145.3.3	looseMarshal	873
6.145.3.4	looseUnmarshal	874
6.145.3.5	tightMarshal1	874
6.145.3.6	tightMarshal2	874
6.145.3.7	tightUnmarshal	875
6.146	activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference	876
6.146.1	Detailed Description	876
6.146.2	Constructor & Destructor Documentation	877
6.146.2.1	BrokerIdMarshaller	877
6.146.2.2	~BrokerIdMarshaller	877
6.146.3	Member Function Documentation	877
6.146.3.1	createObject	877
6.146.3.2	getDataStructureType	877
6.146.3.3	looseMarshal	877
6.146.3.4	looseUnmarshal	878
6.146.3.5	tightMarshal1	878
6.146.3.6	tightMarshal2	878
6.146.3.7	tightUnmarshal	879

6.147	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference	880
6.147.1	Detailed Description	880
6.147.2	Constructor & Destructor Documentation	881
6.147.2.1	BrokerIdMarshaller	881
6.147.2.2	~BrokerIdMarshaller	881
6.147.3	Member Function Documentation	881
6.147.3.1	createObject	881
6.147.3.2	getDataStructureType	881
6.147.3.3	looseMarshal	881
6.147.3.4	looseUnmarshal	882
6.147.3.5	tightMarshal1	882
6.147.3.6	tightMarshal2	882
6.147.3.7	tightUnmarshal	883
6.148	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference	884
6.148.1	Detailed Description	884
6.148.2	Constructor & Destructor Documentation	885
6.148.2.1	BrokerIdMarshaller	885
6.148.2.2	~BrokerIdMarshaller	885
6.148.3	Member Function Documentation	885
6.148.3.1	createObject	885
6.148.3.2	getDataStructureType	885
6.148.3.3	looseMarshal	885
6.148.3.4	looseUnmarshal	886
6.148.3.5	tightMarshal1	886
6.148.3.6	tightMarshal2	886
6.148.3.7	tightUnmarshal	887
6.149	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference	888
6.149.1	Detailed Description	888
6.149.2	Constructor & Destructor Documentation	889
6.149.2.1	BrokerIdMarshaller	889
6.149.2.2	~BrokerIdMarshaller	889
6.149.3	Member Function Documentation	889
6.149.3.1	createObject	889
6.149.3.2	getDataStructureType	889
6.149.3.3	looseMarshal	889
6.149.3.4	looseUnmarshal	890

6.149.3.5 tightMarshal1 . . . . .	890
6.149.3.6 tightMarshal2 . . . . .	890
6.149.3.7 tightUnmarshal . . . . .	891
6.150activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference . . . . .	892
6.150.1 Detailed Description . . . . .	892
6.150.2 Constructor & Destructor Documentation . . . . .	893
6.150.2.1 BrokerIdMarshaller . . . . .	893
6.150.2.2 ~BrokerIdMarshaller . . . . .	893
6.150.3 Member Function Documentation . . . . .	893
6.150.3.1 createObject . . . . .	893
6.150.3.2 getDataStructureType . . . . .	893
6.150.3.3 looseMarshal . . . . .	893
6.150.3.4 looseUnmarshal . . . . .	894
6.150.3.5 tightMarshal1 . . . . .	894
6.150.3.6 tightMarshal2 . . . . .	894
6.150.3.7 tightUnmarshal . . . . .	895
6.151activemq::commands::BrokerInfo Class Reference . . . . .	896
6.151.1 Constructor & Destructor Documentation . . . . .	897
6.151.1.1 BrokerInfo . . . . .	897
6.151.1.2 ~BrokerInfo . . . . .	897
6.151.2 Member Function Documentation . . . . .	897
6.151.2.1 cloneDataStructure . . . . .	897
6.151.2.2 copyDataStructure . . . . .	898
6.151.2.3 equals . . . . .	898
6.151.2.4 getBrokerId . . . . .	899
6.151.2.5 getBrokerId . . . . .	899
6.151.2.6 getBrokerName . . . . .	899
6.151.2.7 getBrokerName . . . . .	899
6.151.2.8 getBrokerUploadUrl . . . . .	899
6.151.2.9 getBrokerUploadUrl . . . . .	899
6.151.2.10getBrokerURL . . . . .	899
6.151.2.11getBrokerURL . . . . .	899
6.151.2.12getConnectionId . . . . .	899
6.151.2.13getDataStructureType . . . . .	899
6.151.2.14getNetworkProperties . . . . .	900
6.151.2.15getNetworkProperties . . . . .	900

6.151.2.16	getPeerBrokerInfos . . . . .	900
6.151.2.17	getPeerBrokerInfos . . . . .	900
6.151.2.18	sBrokerInfo . . . . .	900
6.151.2.19	sDuplexConnection . . . . .	901
6.151.2.20	sFaultTolerantConfiguration . . . . .	901
6.151.2.21	isMasterBroker . . . . .	901
6.151.2.22	sNetworkConnection . . . . .	901
6.151.2.23	sSlaveBroker . . . . .	901
6.151.2.24	setBrokerId . . . . .	901
6.151.2.25	setBrokerName . . . . .	901
6.151.2.26	setBrokerUploadUrl . . . . .	901
6.151.2.27	setBrokerURL . . . . .	901
6.151.2.28	setConnectionId . . . . .	901
6.151.2.29	setDuplexConnection . . . . .	901
6.151.2.30	setFaultTolerantConfiguration . . . . .	901
6.151.2.31	setMasterBroker . . . . .	901
6.151.2.32	setNetworkConnection . . . . .	901
6.151.2.33	setNetworkProperties . . . . .	901
6.151.2.34	setPeerBrokerInfos . . . . .	901
6.151.2.35	setSlaveBroker . . . . .	901
6.151.2.36	toString . . . . .	901
6.151.2.37	visit . . . . .	902
6.151.3	Field Documentation . . . . .	903
6.151.3.1	brokerId . . . . .	903
6.151.3.2	brokerName . . . . .	903
6.151.3.3	brokerUploadUrl . . . . .	903
6.151.3.4	brokerURL . . . . .	903
6.151.3.5	connectionId . . . . .	903
6.151.3.6	duplexConnection . . . . .	903
6.151.3.7	faultTolerantConfiguration . . . . .	903
6.151.3.8	ID_BROKERINFO . . . . .	903
6.151.3.9	masterBroker . . . . .	903
6.151.3.10	networkConnection . . . . .	903
6.151.3.11	networkProperties . . . . .	903
6.151.3.12	peerBrokerInfos . . . . .	903
6.151.3.13	slaveBroker . . . . .	903

6.152activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference	904
6.152.1 Detailed Description	904
6.152.2 Constructor & Destructor Documentation	905
6.152.2.1 BrokerInfoMarshaller	905
6.152.2.2 ~BrokerInfoMarshaller	905
6.152.3 Member Function Documentation	905
6.152.3.1 createObject	905
6.152.3.2 getDataStructureType	905
6.152.3.3 looseMarshal	905
6.152.3.4 looseUnmarshal	906
6.152.3.5 tightMarshal1	906
6.152.3.6 tightMarshal2	906
6.152.3.7 tightUnmarshal	907
6.153activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference	908
6.153.1 Detailed Description	908
6.153.2 Constructor & Destructor Documentation	909
6.153.2.1 BrokerInfoMarshaller	909
6.153.2.2 ~BrokerInfoMarshaller	909
6.153.3 Member Function Documentation	909
6.153.3.1 createObject	909
6.153.3.2 getDataStructureType	909
6.153.3.3 looseMarshal	909
6.153.3.4 looseUnmarshal	910
6.153.3.5 tightMarshal1	910
6.153.3.6 tightMarshal2	910
6.153.3.7 tightUnmarshal	911
6.154activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference	912
6.154.1 Detailed Description	912
6.154.2 Constructor & Destructor Documentation	913
6.154.2.1 BrokerInfoMarshaller	913
6.154.2.2 ~BrokerInfoMarshaller	913
6.154.3 Member Function Documentation	913
6.154.3.1 createObject	913
6.154.3.2 getDataStructureType	913
6.154.3.3 looseMarshal	913
6.154.3.4 looseUnmarshal	914

6.154.3.5 tightMarshal1 . . . . .	914
6.154.3.6 tightMarshal2 . . . . .	914
6.154.3.7 tightUnmarshal . . . . .	915
6.155activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference	916
6.155.1 Detailed Description . . . . .	916
6.155.2 Constructor & Destructor Documentation . . . . .	917
6.155.2.1 BrokerInfoMarshaller . . . . .	917
6.155.2.2 ~BrokerInfoMarshaller . . . . .	917
6.155.3 Member Function Documentation . . . . .	917
6.155.3.1 createObject . . . . .	917
6.155.3.2 getDataStructureType . . . . .	917
6.155.3.3 looseMarshal . . . . .	917
6.155.3.4 looseUnmarshal . . . . .	918
6.155.3.5 tightMarshal1 . . . . .	918
6.155.3.6 tightMarshal2 . . . . .	918
6.155.3.7 tightUnmarshal . . . . .	919
6.156activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class Reference	920
6.156.1 Detailed Description . . . . .	920
6.156.2 Constructor & Destructor Documentation . . . . .	921
6.156.2.1 BrokerInfoMarshaller . . . . .	921
6.156.2.2 ~BrokerInfoMarshaller . . . . .	921
6.156.3 Member Function Documentation . . . . .	921
6.156.3.1 createObject . . . . .	921
6.156.3.2 getDataStructureType . . . . .	921
6.156.3.3 looseMarshal . . . . .	921
6.156.3.4 looseUnmarshal . . . . .	922
6.156.3.5 tightMarshal1 . . . . .	922
6.156.3.6 tightMarshal2 . . . . .	922
6.156.3.7 tightUnmarshal . . . . .	923
6.157activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference	924
6.157.1 Detailed Description . . . . .	924
6.157.2 Constructor & Destructor Documentation . . . . .	925
6.157.2.1 BrokerInfoMarshaller . . . . .	925
6.157.2.2 ~BrokerInfoMarshaller . . . . .	925
6.157.3 Member Function Documentation . . . . .	925
6.157.3.1 createObject . . . . .	925

6.157.3.2	getDataStructureType . . . . .	925
6.157.3.3	looseMarshal . . . . .	925
6.157.3.4	looseUnmarshal . . . . .	926
6.157.3.5	tightMarshal1 . . . . .	926
6.157.3.6	tightMarshal2 . . . . .	926
6.157.3.7	tightUnmarshal . . . . .	927
6.158	decaf::nio::Buffer Class Reference . . . . .	928
6.158.1	Detailed Description . . . . .	929
6.158.2	Constructor & Destructor Documentation . . . . .	930
6.158.2.1	Buffer . . . . .	930
6.158.2.2	Buffer . . . . .	930
6.158.2.3	~Buffer . . . . .	930
6.158.3	Member Function Documentation . . . . .	930
6.158.3.1	capacity . . . . .	930
6.158.3.2	clear . . . . .	930
6.158.3.3	flip . . . . .	931
6.158.3.4	hasRemaining . . . . .	931
6.158.3.5	isReadOnly . . . . .	931
6.158.3.6	limit . . . . .	931
6.158.3.7	limit . . . . .	932
6.158.3.8	mark . . . . .	932
6.158.3.9	position . . . . .	932
6.158.3.10	position . . . . .	932
6.158.3.11	remaining . . . . .	933
6.158.3.12	reset . . . . .	933
6.158.3.13	rewind . . . . .	933
6.158.4	Field Documentation . . . . .	933
6.158.4.1	_capacity . . . . .	933
6.158.4.2	_limit . . . . .	933
6.158.4.3	_mark . . . . .	933
6.158.4.4	_markSet . . . . .	933
6.158.4.5	_position . . . . .	933
6.159	decaf::io::BufferedInputStream Class Reference . . . . .	934
6.159.1	Detailed Description . . . . .	936
6.159.2	Constructor & Destructor Documentation . . . . .	936
6.159.2.1	BufferedInputStream . . . . .	936

6.159.2.2	BufferedInputStream	936
6.159.2.3	~BufferedInputStream	936
6.159.3	Member Function Documentation	936
6.159.3.1	available	936
6.159.3.2	close	937
6.159.3.3	doReadArrayBounded	937
6.159.3.4	doReadByte	937
6.159.3.5	mark	937
6.159.3.6	markSupported	937
6.159.3.7	reset	938
6.159.3.8	skip	938
6.160	decaf::io::BufferedOutputStream Class Reference	940
6.160.1	Detailed Description	940
6.160.2	Constructor & Destructor Documentation	940
6.160.2.1	BufferedOutputStream	940
6.160.2.2	BufferedOutputStream	941
6.160.2.3	~BufferedOutputStream	941
6.160.3	Member Function Documentation	941
6.160.3.1	doWriteArray	941
6.160.3.2	doWriteArrayBounded	941
6.160.3.3	doWriteByte	941
6.160.3.4	flush	941
6.161	decaf::internal::nio::BufferFactory Class Reference	942
6.161.1	Detailed Description	944
6.161.2	Constructor & Destructor Documentation	944
6.161.2.1	~BufferFactory	944
6.161.3	Member Function Documentation	944
6.161.3.1	createByteBuffer	944
6.161.3.2	createByteBuffer	944
6.161.3.3	createByteBuffer	945
6.161.3.4	createCharBuffer	945
6.161.3.5	createCharBuffer	946
6.161.3.6	createCharBuffer	946
6.161.3.7	createDoubleBuffer	946
6.161.3.8	createDoubleBuffer	947
6.161.3.9	createDoubleBuffer	947



6.161.3.10	createFloatBuffer	948
6.161.3.11	createFloatBuffer	948
6.161.3.12	createFloatBuffer	949
6.161.3.13	createIntBuffer	949
6.161.3.14	createIntBuffer	949
6.161.3.15	createIntBuffer	950
6.161.3.16	createLongBuffer	950
6.161.3.17	createLongBuffer	951
6.161.3.18	createLongBuffer	951
6.161.3.19	createShortBuffer	951
6.161.3.20	createShortBuffer	952
6.161.3.21	createShortBuffer	952
6.162	decaf::nio::BufferOverflowException Class Reference	954
6.162.1	Constructor & Destructor Documentation	954
6.162.1.1	BufferOverflowException	954
6.162.1.2	BufferOverflowException	954
6.162.1.3	BufferOverflowException	955
6.162.1.4	BufferOverflowException	955
6.162.1.5	BufferOverflowException	955
6.162.1.6	BufferOverflowException	955
6.162.1.7	~BufferOverflowException	956
6.162.2	Member Function Documentation	956
6.162.2.1	clone	956
6.163	decaf::nio::BufferUnderflowException Class Reference	957
6.163.1	Constructor & Destructor Documentation	957
6.163.1.1	BufferUnderflowException	957
6.163.1.2	BufferUnderflowException	957
6.163.1.3	BufferUnderflowException	958
6.163.1.4	BufferUnderflowException	958
6.163.1.5	BufferUnderflowException	958
6.163.1.6	BufferUnderflowException	958
6.163.1.7	~BufferUnderflowException	959
6.163.2	Member Function Documentation	959
6.163.2.1	clone	959
6.164	decaf::lang::Byte Class Reference	960
6.164.1	Constructor & Destructor Documentation	961

6.164.1.1 Byte . . . . .	961
6.164.1.2 Byte . . . . .	962
6.164.1.3 ~Byte . . . . .	962
6.164.2 Member Function Documentation . . . . .	962
6.164.2.1 byteValue . . . . .	962
6.164.2.2 compareTo . . . . .	962
6.164.2.3 compareTo . . . . .	962
6.164.2.4 decode . . . . .	963
6.164.2.5 doubleValue . . . . .	963
6.164.2.6 equals . . . . .	963
6.164.2.7 equals . . . . .	963
6.164.2.8 float Value . . . . .	964
6.164.2.9 int Value . . . . .	964
6.164.2.10 long Value . . . . .	964
6.164.2.11 operator< . . . . .	964
6.164.2.12 operator< . . . . .	964
6.164.2.13 operator== . . . . .	965
6.164.2.14 operator== . . . . .	965
6.164.2.15 parseByte . . . . .	965
6.164.2.16 parseByte . . . . .	966
6.164.2.17 short Value . . . . .	966
6.164.2.18 oString . . . . .	966
6.164.2.19 oString . . . . .	966
6.164.2.20 valueOf . . . . .	967
6.164.2.21 valueOf . . . . .	967
6.164.2.22 valueOf . . . . .	967
6.164.3 Field Documentation . . . . .	968
6.164.3.1 MAX_VALUE . . . . .	968
6.164.3.2 MIN_VALUE . . . . .	968
6.164.3.3 SIZE . . . . .	968
6.165 decaf::internal::util::ByteArrayAdapter Class Reference . . . . .	969
6.165.1 Detailed Description . . . . .	973
6.165.2 Constructor & Destructor Documentation . . . . .	973
6.165.2.1 ByteArrayAdapter . . . . .	973
6.165.2.2 ByteArrayAdapter . . . . .	973
6.165.2.3 ByteArrayAdapter . . . . .	974

6.165.2.4 ByteArrayAdapter . . . . .	974
6.165.2.5 ByteArrayAdapter . . . . .	975
6.165.2.6 ByteArrayAdapter . . . . .	975
6.165.2.7 ByteArrayAdapter . . . . .	975
6.165.2.8 ByteArrayAdapter . . . . .	976
6.165.2.9 ~ByteArrayAdapter . . . . .	976
6.165.3 Member Function Documentation . . . . .	976
6.165.3.1 clear . . . . .	976
6.165.3.2 get . . . . .	976
6.165.3.3 getByteArray . . . . .	977
6.165.3.4 getCapacity . . . . .	977
6.165.3.5 getChar . . . . .	977
6.165.3.6 getCharArray . . . . .	977
6.165.3.7 getCharCapacity . . . . .	977
6.165.3.8 getDouble . . . . .	978
6.165.3.9 getDoubleArray . . . . .	978
6.165.3.10 getDoubleAt . . . . .	978
6.165.3.11 getDoubleCapacity . . . . .	979
6.165.3.12 getFloat . . . . .	979
6.165.3.13 getFloatArray . . . . .	979
6.165.3.14 getFloatAt . . . . .	979
6.165.3.15 getFloatCapacity . . . . .	980
6.165.3.16 getInt . . . . .	980
6.165.3.17 getIntArray . . . . .	980
6.165.3.18 getIntAt . . . . .	980
6.165.3.19 getIntCapacity . . . . .	981
6.165.3.20 getLong . . . . .	981
6.165.3.21 getLongArray . . . . .	981
6.165.3.22 getLongAt . . . . .	982
6.165.3.23 getLongCapacity . . . . .	982
6.165.3.24 getShort . . . . .	982
6.165.3.25 getShortArray . . . . .	983
6.165.3.26 getShortAt . . . . .	983
6.165.3.27 getShortCapacity . . . . .	983
6.165.3.28 operator[] . . . . .	983
6.165.3.29 operator[] . . . . .	983

6.165.3.30	put	984
6.165.3.31	putChar	984
6.165.3.32	putDouble	985
6.165.3.33	putDoubleAt	985
6.165.3.34	putFloat	985
6.165.3.35	putFloatAt	986
6.165.3.36	putInt	986
6.165.3.37	putIntAt	987
6.165.3.38	putLong	987
6.165.3.39	putLongAt	987
6.165.3.40	putShort	988
6.165.3.41	putShortAt	988
6.165.3.42	read	989
6.165.3.43	resize	989
6.165.3.44	write	989
6.166	decaf::internal::nio::ByteBuffer Class Reference	991
6.166.1	Detailed Description	1001
6.166.2	Constructor & Destructor Documentation	1002
6.166.2.1	ByteBuffer	1002
6.166.2.2	ByteBuffer	1002
6.166.2.3	ByteBuffer	1003
6.166.2.4	ByteBuffer	1003
6.166.2.5	~ByteBuffer	1003
6.166.3	Member Function Documentation	1003
6.166.3.1	array	1003
6.166.3.2	arrayOffset	1004
6.166.3.3	asCharBuffer	1004
6.166.3.4	asDoubleBuffer	1004
6.166.3.5	asFloatBuffer	1005
6.166.3.6	asIntBuffer	1005
6.166.3.7	asLongBuffer	1006
6.166.3.8	asReadOnlyBuffer	1006
6.166.3.9	asShortBuffer	1006
6.166.3.10	compact	1007
6.166.3.11	duplicate	1007
6.166.3.12	get	1007

6.166.3.13	get . . . . .	1008
6.166.3.14	getChar . . . . .	1008
6.166.3.15	getChar . . . . .	1009
6.166.3.16	getDouble . . . . .	1009
6.166.3.17	getDouble . . . . .	1009
6.166.3.18	getFloat . . . . .	1010
6.166.3.19	getFloat . . . . .	1010
6.166.3.20	getInt . . . . .	1010
6.166.3.21	getInt . . . . .	1011
6.166.3.22	getLong . . . . .	1011
6.166.3.23	getLong . . . . .	1011
6.166.3.24	getShort . . . . .	1012
6.166.3.25	getShort . . . . .	1012
6.166.3.26	hasArray . . . . .	1012
6.166.3.27	isReadOnly . . . . .	1013
6.166.3.28	put . . . . .	1013
6.166.3.29	put . . . . .	1013
6.166.3.30	putChar . . . . .	1014
6.166.3.31	putChar . . . . .	1014
6.166.3.32	putDouble . . . . .	1014
6.166.3.33	putDouble . . . . .	1015
6.166.3.34	putFloat . . . . .	1015
6.166.3.35	putFloat . . . . .	1016
6.166.3.36	putInt . . . . .	1016
6.166.3.37	putInt . . . . .	1017
6.166.3.38	putLong . . . . .	1017
6.166.3.39	putLong . . . . .	1017
6.166.3.40	putShort . . . . .	1018
6.166.3.41	putShort . . . . .	1018
6.166.3.42	setReadOnly . . . . .	1019
6.166.3.43	lice . . . . .	1019
6.167	decaf::io::ByteArrayInputStream Class Reference . . . . .	1020
6.167.1	Detailed Description . . . . .	1022
6.167.2	Constructor & Destructor Documentation . . . . .	1022
6.167.2.1	ByteArrayInputStream . . . . .	1022
6.167.2.2	ByteArrayInputStream . . . . .	1022

6.167.2.3	ByteArrayInputStream	1023
6.167.2.4	ByteArrayInputStream	1023
6.167.2.5	~ByteArrayInputStream	1023
6.167.3	Member Function Documentation	1023
6.167.3.1	available	1023
6.167.3.2	doReadArrayBounded	1024
6.167.3.3	doReadByte	1024
6.167.3.4	mark	1024
6.167.3.5	markSupported	1024
6.167.3.6	reset	1025
6.167.3.7	setByteArray	1025
6.167.3.8	setByteArray	1026
6.167.3.9	setByteArray	1026
6.167.3.10	skip	1026
6.168	decaf::io::ByteArrayOutputStream Class Reference	1028
6.168.1	Constructor & Destructor Documentation	1028
6.168.1.1	ByteArrayOutputStream	1028
6.168.1.2	ByteArrayOutputStream	1029
6.168.1.3	~ByteArrayOutputStream	1029
6.168.2	Member Function Documentation	1029
6.168.2.1	doWriteArrayBounded	1029
6.168.2.2	doWriteByte	1029
6.168.2.3	reset	1029
6.168.2.4	size	1029
6.168.2.5	toArray	1030
6.168.2.6	toString	1030
6.168.2.7	writeTo	1030
6.169	decaf::nio::ByteBuffer Class Reference	1031
6.169.1	Detailed Description	1035
6.169.2	Constructor & Destructor Documentation	1036
6.169.2.1	ByteBuffer	1036
6.169.2.2	~ByteBuffer	1036
6.169.3	Member Function Documentation	1036
6.169.3.1	allocate	1036
6.169.3.2	array	1037
6.169.3.3	arrayOffset	1037

6.169.3.4 asCharBuffer . . . . .	1037
6.169.3.5 asDoubleBuffer . . . . .	1038
6.169.3.6 asFloatBuffer . . . . .	1038
6.169.3.7 asIntBuffer . . . . .	1038
6.169.3.8 asLongBuffer . . . . .	1039
6.169.3.9 asReadOnlyBuffer . . . . .	1039
6.169.3.10 asShortBuffer . . . . .	1039
6.169.3.11 compact . . . . .	1040
6.169.3.12 compareTo . . . . .	1040
6.169.3.13 duplicate . . . . .	1040
6.169.3.14 equals . . . . .	1040
6.169.3.15 get . . . . .	1040
6.169.3.16 get . . . . .	1041
6.169.3.17 get . . . . .	1041
6.169.3.18 get . . . . .	1042
6.169.3.19 getChar . . . . .	1042
6.169.3.20 getChar . . . . .	1042
6.169.3.21 getDouble . . . . .	1043
6.169.3.22 getDouble . . . . .	1043
6.169.3.23 getFloat . . . . .	1043
6.169.3.24 getFloat . . . . .	1044
6.169.3.25 getInt . . . . .	1044
6.169.3.26 getInt . . . . .	1044
6.169.3.27 getLong . . . . .	1045
6.169.3.28 getLong . . . . .	1045
6.169.3.29 getShort . . . . .	1045
6.169.3.30 getShort . . . . .	1046
6.169.3.31 hasArray . . . . .	1046
6.169.3.32 isReadOnly . . . . .	1046
6.169.3.33 operator< . . . . .	1046
6.169.3.34 operator== . . . . .	1046
6.169.3.35 put . . . . .	1046
6.169.3.36 put . . . . .	1047
6.169.3.37 put . . . . .	1047
6.169.3.38 put . . . . .	1048
6.169.3.39 put . . . . .	1048

6.169.3.40	putChar . . . . .	1049
6.169.3.41	putChar . . . . .	1049
6.169.3.42	putDouble . . . . .	1050
6.169.3.43	putDouble . . . . .	1050
6.169.3.44	putFloat . . . . .	1050
6.169.3.45	putFloat . . . . .	1051
6.169.3.46	putInt . . . . .	1051
6.169.3.47	putInt . . . . .	1052
6.169.3.48	putLong . . . . .	1052
6.169.3.49	putLong . . . . .	1052
6.169.3.50	putShort . . . . .	1053
6.169.3.51	putShort . . . . .	1053
6.169.3.52	slice . . . . .	1054
6.169.3.53	toString . . . . .	1054
6.169.3.54	wrap . . . . .	1054
6.169.3.55	wrap . . . . .	1054
6.170	cms::BytesMessage Class Reference . . . . .	1056
6.170.1	Detailed Description . . . . .	1058
6.170.2	Constructor & Destructor Documentation . . . . .	1059
6.170.2.1	~BytesMessage . . . . .	1059
6.170.3	Member Function Documentation . . . . .	1059
6.170.3.1	clone . . . . .	1059
6.170.3.2	getBodyBytes . . . . .	1059
6.170.3.3	getBodyLength . . . . .	1060
6.170.3.4	readBoolean . . . . .	1060
6.170.3.5	readByte . . . . .	1060
6.170.3.6	readBytes . . . . .	1061
6.170.3.7	readBytes . . . . .	1061
6.170.3.8	readChar . . . . .	1062
6.170.3.9	readDouble . . . . .	1062
6.170.3.10	readFloat . . . . .	1063
6.170.3.11	readInt . . . . .	1063
6.170.3.12	readLong . . . . .	1063
6.170.3.13	readShort . . . . .	1064
6.170.3.14	readString . . . . .	1064
6.170.3.15	readUnsignedShort . . . . .	1064



6.170.3.16	readUTF . . . . .	1065
6.170.3.17	reset . . . . .	1065
6.170.3.18	setBodyBytes . . . . .	1065
6.170.3.19	writeBoolean . . . . .	1066
6.170.3.20	writeByte . . . . .	1066
6.170.3.21	writeBytes . . . . .	1066
6.170.3.22	writeBytes . . . . .	1067
6.170.3.23	writeChar . . . . .	1067
6.170.3.24	writeDouble . . . . .	1067
6.170.3.25	writeFloat . . . . .	1068
6.170.3.26	writeInt . . . . .	1068
6.170.3.27	writeLong . . . . .	1068
6.170.3.28	writeShort . . . . .	1069
6.170.3.29	writeString . . . . .	1069
6.170.3.30	writeUnsignedShort . . . . .	1069
6.170.3.31	writeUTF . . . . .	1070
6.171	activemq::cmsutil::CachedConsumer Class Reference . . . . .	1071
6.171.1	Detailed Description . . . . .	1071
6.171.2	Constructor & Destructor Documentation . . . . .	1072
6.171.2.1	CachedConsumer . . . . .	1072
6.171.2.2	CachedConsumer . . . . .	1072
6.171.2.3	~CachedConsumer . . . . .	1072
6.171.3	Member Function Documentation . . . . .	1072
6.171.3.1	close . . . . .	1072
6.171.3.2	getMessageListener . . . . .	1072
6.171.3.3	getMessageSelector . . . . .	1072
6.171.3.4	operator= . . . . .	1073
6.171.3.5	receive . . . . .	1073
6.171.3.6	receive . . . . .	1073
6.171.3.7	receiveNoWait . . . . .	1073
6.171.3.8	setMessageListener . . . . .	1073
6.172	activemq::cmsutil::CachedProducer Class Reference . . . . .	1075
6.172.1	Detailed Description . . . . .	1076
6.172.2	Constructor & Destructor Documentation . . . . .	1076
6.172.2.1	CachedProducer . . . . .	1076
6.172.2.2	CachedProducer . . . . .	1076

6.172.2.3 ~CachedProducer . . . . .	1076
6.172.3 Member Function Documentation . . . . .	1076
6.172.3.1 close . . . . .	1076
6.172.3.2 getDeliveryMode . . . . .	1076
6.172.3.3 getDisableMessageID . . . . .	1077
6.172.3.4 getDisableMessageTimeStamp . . . . .	1077
6.172.3.5 getPriority . . . . .	1077
6.172.3.6 getTimeToLive . . . . .	1078
6.172.3.7 operator= . . . . .	1078
6.172.3.8 send . . . . .	1078
6.172.3.9 send . . . . .	1078
6.172.3.10 send . . . . .	1079
6.172.3.11 send . . . . .	1079
6.172.3.12 setDeliveryMode . . . . .	1080
6.172.3.13 setDisableMessageID . . . . .	1080
6.172.3.14 setDisableMessageTimeStamp . . . . .	1080
6.172.3.15 setPriority . . . . .	1081
6.172.3.16 setTimeToLive . . . . .	1081
6.173 decaf::util::concurrent::Callable< V > Class Template Reference . . . . .	1082
6.173.1 Detailed Description . . . . .	1082
6.173.2 Constructor & Destructor Documentation . . . . .	1082
6.173.2.1 ~Callable . . . . .	1082
6.173.3 Member Function Documentation . . . . .	1082
6.173.3.1 call . . . . .	1082
6.174 decaf::util::concurrent::CancellationException Class Reference . . . . .	1083
6.174.1 Constructor & Destructor Documentation . . . . .	1083
6.174.1.1 CancellationException . . . . .	1083
6.174.1.2 CancellationException . . . . .	1083
6.174.1.3 CancellationException . . . . .	1084
6.174.1.4 CancellationException . . . . .	1084
6.174.1.5 CancellationException . . . . .	1084
6.174.1.6 CancellationException . . . . .	1084
6.174.1.7 ~CancellationException . . . . .	1085
6.174.2 Member Function Documentation . . . . .	1085
6.174.2.1 clone . . . . .	1085
6.175 decaf::security::cert::Certificate Class Reference . . . . .	1086

6.175.1 Detailed Description . . . . .	1086
6.175.2 Constructor & Destructor Documentation . . . . .	1087
6.175.2.1 ~Certificate . . . . .	1087
6.175.3 Member Function Documentation . . . . .	1087
6.175.3.1 equals . . . . .	1087
6.175.3.2 getEncoded . . . . .	1087
6.175.3.3 getPublicKey . . . . .	1087
6.175.3.4 getPublicKey . . . . .	1087
6.175.3.5 getType . . . . .	1088
6.175.3.6 toString . . . . .	1088
6.175.3.7 verify . . . . .	1088
6.175.3.8 verify . . . . .	1088
6.176decaf::security::cert::CertificateEncodingException Class Reference . . . . .	1090
6.176.1 Constructor & Destructor Documentation . . . . .	1090
6.176.1.1 CertificateEncodingException . . . . .	1090
6.176.1.2 CertificateEncodingException . . . . .	1090
6.176.1.3 CertificateEncodingException . . . . .	1090
6.176.1.4 CertificateEncodingException . . . . .	1091
6.176.1.5 ~CertificateEncodingException . . . . .	1091
6.176.2 Member Function Documentation . . . . .	1091
6.176.2.1 clone . . . . .	1091
6.177decaf::security::cert::CertificateException Class Reference . . . . .	1092
6.177.1 Constructor & Destructor Documentation . . . . .	1092
6.177.1.1 CertificateException . . . . .	1092
6.177.1.2 CertificateException . . . . .	1092
6.177.1.3 CertificateException . . . . .	1092
6.177.1.4 CertificateException . . . . .	1093
6.177.1.5 ~CertificateException . . . . .	1093
6.177.2 Member Function Documentation . . . . .	1093
6.177.2.1 clone . . . . .	1093
6.178decaf::security::cert::CertificateExpiredException Class Reference . . . . .	1094
6.178.1 Constructor & Destructor Documentation . . . . .	1094
6.178.1.1 CertificateExpiredException . . . . .	1094
6.178.1.2 CertificateExpiredException . . . . .	1094
6.178.1.3 CertificateExpiredException . . . . .	1094
6.178.1.4 CertificateExpiredException . . . . .	1095

6.178.1.5 ~CertificateExpiredException . . . . .	1095
6.178.2 Member Function Documentation . . . . .	1095
6.178.2.1 clone . . . . .	1095
6.179decaf::security::cert::CertificateNotYetValidException Class Reference . . . . .	1096
6.179.1 Constructor & Destructor Documentation . . . . .	1096
6.179.1.1 CertificateNotYetValidException . . . . .	1096
6.179.1.2 CertificateNotYetValidException . . . . .	1096
6.179.1.3 CertificateNotYetValidException . . . . .	1096
6.179.1.4 CertificateNotYetValidException . . . . .	1097
6.179.1.5 ~CertificateNotYetValidException . . . . .	1097
6.179.2 Member Function Documentation . . . . .	1097
6.179.2.1 clone . . . . .	1097
6.180decaf::security::cert::CertificateParsingException Class Reference . . . . .	1098
6.180.1 Constructor & Destructor Documentation . . . . .	1098
6.180.1.1 CertificateParsingException . . . . .	1098
6.180.1.2 CertificateParsingException . . . . .	1098
6.180.1.3 CertificateParsingException . . . . .	1098
6.180.1.4 CertificateParsingException . . . . .	1099
6.180.1.5 ~CertificateParsingException . . . . .	1099
6.180.2 Member Function Documentation . . . . .	1099
6.180.2.1 clone . . . . .	1099
6.181decaf::lang::Character Class Reference . . . . .	1100
6.181.1 Constructor & Destructor Documentation . . . . .	1102
6.181.1.1 Character . . . . .	1102
6.181.2 Member Function Documentation . . . . .	1102
6.181.2.1 byteValue . . . . .	1102
6.181.2.2 compareTo . . . . .	1102
6.181.2.3 compareTo . . . . .	1102
6.181.2.4 digit . . . . .	1103
6.181.2.5 doubleValue . . . . .	1103
6.181.2.6 equals . . . . .	1103
6.181.2.7 equals . . . . .	1103
6.181.2.8 floatValue . . . . .	1103
6.181.2.9 intValue . . . . .	1104
6.181.2.10sDigit . . . . .	1104
6.181.2.11sISOControl . . . . .	1104

6.181.2.12sLetter . . . . .	1104
6.181.2.13sLetterOrDigit . . . . .	1104
6.181.2.14sLowerCase . . . . .	1104
6.181.2.15sUpperCase . . . . .	1104
6.181.2.16sWhitespace . . . . .	1104
6.181.2.17ongValue . . . . .	1105
6.181.2.18operator< . . . . .	1105
6.181.2.19operator< . . . . .	1105
6.181.2.20operator== . . . . .	1105
6.181.2.21operator== . . . . .	1106
6.181.2.22shortValue . . . . .	1106
6.181.2.23oString . . . . .	1106
6.181.2.24valueOf . . . . .	1106
6.181.3 Field Documentation . . . . .	1106
6.181.3.1 MAX_RADIX . . . . .	1106
6.181.3.2 MAX_VALUE . . . . .	1106
6.181.3.3 MIN_RADIX . . . . .	1107
6.181.3.4 MIN_VALUE . . . . .	1107
6.181.3.5 SIZE . . . . .	1107
6.182decaf::nio::CharArrayBuffer Class Reference . . . . .	1108
6.182.1 Constructor & Destructor Documentation . . . . .	1111
6.182.1.1 CharArrayBuffer . . . . .	1111
6.182.1.2 CharArrayBuffer . . . . .	1112
6.182.1.3 CharArrayBuffer . . . . .	1112
6.182.1.4 CharArrayBuffer . . . . .	1113
6.182.1.5 ~CharArrayBuffer . . . . .	1113
6.182.2 Member Function Documentation . . . . .	1113
6.182.2.1 array . . . . .	1113
6.182.2.2 arrayOffset . . . . .	1113
6.182.2.3 asReadOnlyBuffer . . . . .	1114
6.182.2.4 compact . . . . .	1114
6.182.2.5 duplicate . . . . .	1114
6.182.2.6 get . . . . .	1115
6.182.2.7 get . . . . .	1115
6.182.2.8 hasArray . . . . .	1115
6.182.2.9 isReadOnly . . . . .	1116

6.182.2.10put . . . . .	1116
6.182.2.11put . . . . .	1116
6.182.2.12setReadOnly . . . . .	1117
6.182.2.13lice . . . . .	1117
6.182.2.14subSequence . . . . .	1117
6.182.3 Field Documentation . . . . .	1118
6.182.3.1 _array . . . . .	1118
6.182.3.2 length . . . . .	1118
6.182.3.3 offset . . . . .	1118
6.182.3.4 readOnly . . . . .	1118
6.183decaf::nio::CharBuffer Class Reference . . . . .	1119
6.183.1 Detailed Description . . . . .	1122
6.183.2 Constructor & Destructor Documentation . . . . .	1122
6.183.2.1 CharBuffer . . . . .	1122
6.183.2.2 ~CharBuffer . . . . .	1122
6.183.3 Member Function Documentation . . . . .	1122
6.183.3.1 allocate . . . . .	1122
6.183.3.2 append . . . . .	1123
6.183.3.3 append . . . . .	1123
6.183.3.4 append . . . . .	1124
6.183.3.5 array . . . . .	1124
6.183.3.6 arrayOffset . . . . .	1124
6.183.3.7 asReadOnlyBuffer . . . . .	1125
6.183.3.8 charAt . . . . .	1125
6.183.3.9 compact . . . . .	1125
6.183.3.10compareTo . . . . .	1126
6.183.3.11duplicate . . . . .	1126
6.183.3.12equals . . . . .	1126
6.183.3.13get . . . . .	1126
6.183.3.14get . . . . .	1127
6.183.3.15get . . . . .	1127
6.183.3.16get . . . . .	1127
6.183.3.17hasArray . . . . .	1128
6.183.3.18length . . . . .	1128
6.183.3.19operator< . . . . .	1128
6.183.3.20operator== . . . . .	1128

6.183.3.21put . . . . .	1128
6.183.3.22put . . . . .	1129
6.183.3.23put . . . . .	1129
6.183.3.24put . . . . .	1130
6.183.3.25put . . . . .	1130
6.183.3.26put . . . . .	1130
6.183.3.27put . . . . .	1131
6.183.3.28read . . . . .	1131
6.183.3.29lice . . . . .	1132
6.183.3.30ubSequence . . . . .	1132
6.183.3.31toString . . . . .	1133
6.183.3.32wrap . . . . .	1133
6.183.3.33wrap . . . . .	1133
6.184decaf::lang::CharSequence Class Reference . . . . .	1135
6.184.1 Detailed Description . . . . .	1135
6.184.2 Constructor & Destructor Documentation . . . . .	1135
6.184.2.1 ~CharSequence . . . . .	1135
6.184.3 Member Function Documentation . . . . .	1135
6.184.3.1 charAt . . . . .	1135
6.184.3.2 length . . . . .	1136
6.184.3.3 subSequence . . . . .	1136
6.184.3.4 toString . . . . .	1136
6.185decaf::util::zip::CheckedInputStream Class Reference . . . . .	1137
6.185.1 Detailed Description . . . . .	1137
6.185.2 Constructor & Destructor Documentation . . . . .	1138
6.185.2.1 CheckedInputStream . . . . .	1138
6.185.2.2 ~CheckedInputStream . . . . .	1138
6.185.3 Member Function Documentation . . . . .	1138
6.185.3.1 doReadArrayBounded . . . . .	1138
6.185.3.2 doReadByte . . . . .	1138
6.185.3.3 getChecksum . . . . .	1138
6.185.3.4 skip . . . . .	1139
6.186decaf::util::zip::CheckedOutputStream Class Reference . . . . .	1140
6.186.1 Detailed Description . . . . .	1140
6.186.2 Constructor & Destructor Documentation . . . . .	1140
6.186.2.1 CheckedOutputStream . . . . .	1140

6.186.2.2 <code>~CheckedOutputStream</code> . . . . .	1141
6.186.3 Member Function Documentation . . . . .	1141
6.186.3.1 <code>doWriteArrayBounded</code> . . . . .	1141
6.186.3.2 <code>doWriteByte</code> . . . . .	1141
6.186.3.3 <code>getChecksum</code> . . . . .	1141
6.187 <code>decaf::util::zip::Checksum</code> Class Reference . . . . .	1142
6.187.1 Detailed Description . . . . .	1142
6.187.2 Constructor & Destructor Documentation . . . . .	1142
6.187.2.1 <code>~Checksum</code> . . . . .	1142
6.187.3 Member Function Documentation . . . . .	1142
6.187.3.1 <code>getValue</code> . . . . .	1142
6.187.3.2 <code>reset</code> . . . . .	1143
6.187.3.3 <code>update</code> . . . . .	1143
6.187.3.4 <code>update</code> . . . . .	1143
6.187.3.5 <code>update</code> . . . . .	1143
6.187.3.6 <code>update</code> . . . . .	1144
6.188 <code>decaf::lang::exceptions::ClassCastException</code> Class Reference . . . . .	1145
6.188.1 Constructor & Destructor Documentation . . . . .	1145
6.188.1.1 <code>ClassCastException</code> . . . . .	1145
6.188.1.2 <code>ClassCastException</code> . . . . .	1145
6.188.1.3 <code>ClassCastException</code> . . . . .	1146
6.188.1.4 <code>ClassCastException</code> . . . . .	1146
6.188.1.5 <code>ClassCastException</code> . . . . .	1146
6.188.1.6 <code>ClassCastException</code> . . . . .	1146
6.188.1.7 <code>~ClassCastException</code> . . . . .	1147
6.188.2 Member Function Documentation . . . . .	1147
6.188.2.1 <code>clone</code> . . . . .	1147
6.189 <code>cms::Closeable</code> Class Reference . . . . .	1148
6.189.1 Detailed Description . . . . .	1148
6.189.2 Constructor & Destructor Documentation . . . . .	1148
6.189.2.1 <code>~Closeable</code> . . . . .	1148
6.189.3 Member Function Documentation . . . . .	1148
6.189.3.1 <code>close</code> . . . . .	1148
6.190 <code>decaf::io::Closeable</code> Class Reference . . . . .	1149
6.190.1 Detailed Description . . . . .	1149
6.190.2 Constructor & Destructor Documentation . . . . .	1149



6.190.2.1 ~Closeable . . . . .	1149
6.190.3 Member Function Documentation . . . . .	1149
6.190.3.1 close . . . . .	1149
6.191activemq::transport::failover::CloseTransportsTask Class Reference . . . . .	1151
6.191.1 Constructor & Destructor Documentation . . . . .	1151
6.191.1.1 CloseTransportsTask . . . . .	1151
6.191.1.2 ~CloseTransportsTask . . . . .	1151
6.191.2 Member Function Documentation . . . . .	1151
6.191.2.1 add . . . . .	1151
6.191.2.2 isPending . . . . .	1151
6.191.2.3 iterate . . . . .	1152
6.192activemq::cmsutil::CmsAccessor Class Reference . . . . .	1153
6.192.1 Detailed Description . . . . .	1154
6.192.2 Constructor & Destructor Documentation . . . . .	1154
6.192.2.1 CmsAccessor . . . . .	1154
6.192.2.2 CmsAccessor . . . . .	1154
6.192.2.3 ~CmsAccessor . . . . .	1154
6.192.3 Member Function Documentation . . . . .	1154
6.192.3.1 checkConnectionFactory . . . . .	1154
6.192.3.2 createConnection . . . . .	1154
6.192.3.3 createSession . . . . .	1155
6.192.3.4 destroy . . . . .	1155
6.192.3.5 getConnectionFactory . . . . .	1155
6.192.3.6 getConnectionFactory . . . . .	1155
6.192.3.7 getResourceLifecycleManager . . . . .	1155
6.192.3.8 getResourceLifecycleManager . . . . .	1155
6.192.3.9 getSessionAcknowledgeMode . . . . .	1155
6.192.3.10init . . . . .	1156
6.192.3.11operator= . . . . .	1156
6.192.3.12setConnectionFactory . . . . .	1156
6.192.3.13setSessionAcknowledgeMode . . . . .	1156
6.193activemq::cmsutil::CmsDestinationAccessor Class Reference . . . . .	1157
6.193.1 Detailed Description . . . . .	1157
6.193.2 Constructor & Destructor Documentation . . . . .	1158
6.193.2.1 CmsDestinationAccessor . . . . .	1158
6.193.2.2 CmsDestinationAccessor . . . . .	1158

6.193.2.3 ~CmsDestinationAccessor . . . . .	1158
6.193.3 Member Function Documentation . . . . .	1158
6.193.3.1 checkDestinationResolver . . . . .	1158
6.193.3.2 destroy . . . . .	1158
6.193.3.3 getDestinationResolver . . . . .	1158
6.193.3.4 getDestinationResolver . . . . .	1158
6.193.3.5 init . . . . .	1158
6.193.3.6 isPubSubDomain . . . . .	1159
6.193.3.7 operator= . . . . .	1159
6.193.3.8 resolveDestinationName . . . . .	1159
6.193.3.9 setDestinationResolver . . . . .	1159
6.193.3.10 setPubSubDomain . . . . .	1159
6.194 cms::CMSException Class Reference . . . . .	1160
6.194.1 Detailed Description . . . . .	1160
6.194.2 Constructor & Destructor Documentation . . . . .	1161
6.194.2.1 CMSException . . . . .	1161
6.194.2.2 CMSException . . . . .	1161
6.194.2.3 CMSException . . . . .	1161
6.194.2.4 CMSException . . . . .	1161
6.194.2.5 ~CMSException . . . . .	1161
6.194.3 Member Function Documentation . . . . .	1161
6.194.3.1 getCause . . . . .	1161
6.194.3.2 getMessage . . . . .	1161
6.194.3.3 getStackTrace . . . . .	1161
6.194.3.4 getStackTraceString . . . . .	1162
6.194.3.5 printStackTrace . . . . .	1162
6.194.3.6 printStackTrace . . . . .	1162
6.194.3.7 setMark . . . . .	1162
6.194.3.8 what . . . . .	1162
6.195 activemq::util::CMSExceptionSupport Class Reference . . . . .	1163
6.195.1 Constructor & Destructor Documentation . . . . .	1163
6.195.1.1 ~CMSExceptionSupport . . . . .	1163
6.195.2 Member Function Documentation . . . . .	1163
6.195.2.1 create . . . . .	1163
6.195.2.2 create . . . . .	1163
6.195.2.3 createMessageEOFException . . . . .	1163

6.195.2.4 createMessageFormatException . . . . .	1163
6.196cms::CMSProperties Class Reference . . . . .	1165
6.196.1 Detailed Description . . . . .	1166
6.196.2 Constructor & Destructor Documentation . . . . .	1166
6.196.2.1 ~CMSProperties . . . . .	1166
6.196.3 Member Function Documentation . . . . .	1166
6.196.3.1 clear . . . . .	1166
6.196.3.2 clone . . . . .	1166
6.196.3.3 copy . . . . .	1166
6.196.3.4 getProperty . . . . .	1166
6.196.3.5 getProperty . . . . .	1167
6.196.3.6 hasProperty . . . . .	1167
6.196.3.7 isEmpty . . . . .	1167
6.196.3.8 remove . . . . .	1167
6.196.3.9 setProperty . . . . .	1168
6.196.3.10oArray . . . . .	1168
6.196.3.11toString . . . . .	1168
6.197cms::CMSSecurityException Class Reference . . . . .	1169
6.197.1 Detailed Description . . . . .	1169
6.197.2 Constructor & Destructor Documentation . . . . .	1169
6.197.2.1 CMSSecurityException . . . . .	1169
6.197.2.2 CMSSecurityException . . . . .	1169
6.197.2.3 CMSSecurityException . . . . .	1169
6.197.2.4 CMSSecurityException . . . . .	1169
6.197.2.5 ~CMSSecurityException . . . . .	1169
6.198activemq::cmsutil::CmsTemplate Class Reference . . . . .	1170
6.198.1 Detailed Description . . . . .	1173
6.198.2 Constructor & Destructor Documentation . . . . .	1173
6.198.2.1 CmsTemplate . . . . .	1173
6.198.2.2 CmsTemplate . . . . .	1173
6.198.2.3 CmsTemplate . . . . .	1173
6.198.2.4 ~CmsTemplate . . . . .	1173
6.198.3 Member Function Documentation . . . . .	1173
6.198.3.1 destroy . . . . .	1173
6.198.3.2 execute . . . . .	1174
6.198.3.3 execute . . . . .	1174

6.198.3.4	execute	1174
6.198.3.5	execute	1174
6.198.3.6	getDefaultDestination	1175
6.198.3.7	getDefaultDestination	1175
6.198.3.8	getDefaultDestinationName	1175
6.198.3.9	getDeliveryMode	1175
6.198.3.10	getPriority	1175
6.198.3.11	getReceiveTimeout	1176
6.198.3.12	getTimeToLive	1176
6.198.3.13	init	1176
6.198.3.14	isExplicitQosEnabled	1176
6.198.3.15	isMessageIdEnabled	1176
6.198.3.16	isMessageTimestampEnabled	1176
6.198.3.17	isNoLocal	1176
6.198.3.18	operator=	1176
6.198.3.19	receive	1176
6.198.3.20	receive	1177
6.198.3.21	receive	1177
6.198.3.22	receiveSelected	1177
6.198.3.23	receiveSelected	1178
6.198.3.24	receiveSelected	1178
6.198.3.25	send	1178
6.198.3.26	send	1179
6.198.3.27	send	1179
6.198.3.28	setDefaultDestination	1179
6.198.3.29	setDefaultDestinationName	1180
6.198.3.30	setDeliveryMode	1180
6.198.3.31	setDeliveryPersistent	1180
6.198.3.32	setExplicitQosEnabled	1180
6.198.3.33	setMessageIdEnabled	1181
6.198.3.34	setMessageTimestampEnabled	1181
6.198.3.35	setNoLocal	1181
6.198.3.36	setPriority	1181
6.198.3.37	setPubSubDomain	1181
6.198.3.38	setReceiveTimeout	1181
6.198.3.39	setTimeToLive	1181

6.198.4 Friends And Related Function Documentation . . . . .	1182
6.198.4.1 ProducerExecutor . . . . .	1182
6.198.4.2 ReceiveExecutor . . . . .	1182
6.198.4.3 ResolveProducerExecutor . . . . .	1182
6.198.4.4 ResolveReceiveExecutor . . . . .	1182
6.198.4.5 SendExecutor . . . . .	1182
6.198.5 Field Documentation . . . . .	1182
6.198.5.1 DEFAULT_PRIORITY . . . . .	1182
6.198.5.2 DEFAULT_TIME_TO_LIVE . . . . .	1182
6.198.5.3 RECEIVE_TIMEOUT_INDEFINITE_WAIT . . . . .	1182
6.198.5.4 RECEIVE_TIMEOUT_NO_WAIT . . . . .	1182
6.199 code Struct Reference . . . . .	1183
6.199.1 Field Documentation . . . . .	1183
6.199.1.1 bits . . . . .	1183
6.199.1.2 op . . . . .	1183
6.199.1.3 val . . . . .	1183
6.200 decaf::util::Collection< E > Class Template Reference . . . . .	1184
6.200.1 Detailed Description . . . . .	1185
6.200.2 Constructor & Destructor Documentation . . . . .	1186
6.200.2.1 ~Collection . . . . .	1186
6.200.3 Member Function Documentation . . . . .	1186
6.200.3.1 add . . . . .	1186
6.200.3.2 addAll . . . . .	1187
6.200.3.3 clear . . . . .	1187
6.200.3.4 contains . . . . .	1188
6.200.3.5 containsAll . . . . .	1189
6.200.3.6 equals . . . . .	1189
6.200.3.7 isEmpty . . . . .	1189
6.200.3.8 remove . . . . .	1190
6.200.3.9 removeAll . . . . .	1191
6.200.3.10 retainAll . . . . .	1191
6.200.3.11 size . . . . .	1192
6.200.3.12 toArray . . . . .	1192
6.201 activemq::commands::Command Class Reference . . . . .	1194
6.201.1 Constructor & Destructor Documentation . . . . .	1195
6.201.1.1 ~Command . . . . .	1195

6.201.2 Member Function Documentation . . . . .	1195
6.201.2.1 getCommandId . . . . .	1195
6.201.2.2 isBrokerInfo . . . . .	1195
6.201.2.3 isConnectionInfo . . . . .	1195
6.201.2.4 isConsumerInfo . . . . .	1195
6.201.2.5 isKeepAliveInfo . . . . .	1195
6.201.2.6 isMessage . . . . .	1195
6.201.2.7 isMessageAck . . . . .	1196
6.201.2.8 isMessageDispatch . . . . .	1196
6.201.2.9 isMessageDispatchNotification . . . . .	1196
6.201.2.10sProducerAck . . . . .	1196
6.201.2.11sProducerInfo . . . . .	1196
6.201.2.12sRemoveInfo . . . . .	1196
6.201.2.13sRemoveSubscriptionInfo . . . . .	1196
6.201.2.14sResponse . . . . .	1196
6.201.2.15sResponseRequired . . . . .	1197
6.201.2.16sShutdownInfo . . . . .	1197
6.201.2.17sTransactionInfo . . . . .	1197
6.201.2.18sWireFormatInfo . . . . .	1197
6.201.2.19setCommandId . . . . .	1197
6.201.2.20setResponseRequired . . . . .	1197
6.201.2.21toString . . . . .	1198
6.201.2.22visit . . . . .	1198
6.202activemq::state::CommandVisitor Class Reference . . . . .	1200
6.202.1 Detailed Description . . . . .	1201
6.202.2 Constructor & Destructor Documentation . . . . .	1202
6.202.2.1 ~CommandVisitor . . . . .	1202
6.202.3 Member Function Documentation . . . . .	1202
6.202.3.1 processBeginTransaction . . . . .	1202
6.202.3.2 processBrokerError . . . . .	1202
6.202.3.3 processBrokerInfo . . . . .	1202
6.202.3.4 processCommitTransactionOnePhase . . . . .	1202
6.202.3.5 processCommitTransactionTwoPhase . . . . .	1202
6.202.3.6 processConnectionControl . . . . .	1203
6.202.3.7 processConnectionError . . . . .	1203
6.202.3.8 processConnectionInfo . . . . .	1203

6.202.3.9 processConsumerControl . . . . .	1203
6.202.3.10 processConsumerInfo . . . . .	1203
6.202.3.11 processControlCommand . . . . .	1203
6.202.3.12 processDestinationInfo . . . . .	1203
6.202.3.13 processEndTransaction . . . . .	1203
6.202.3.14 processFlushCommand . . . . .	1204
6.202.3.15 processForgetTransaction . . . . .	1204
6.202.3.16 processKeepAliveInfo . . . . .	1204
6.202.3.17 processMessage . . . . .	1204
6.202.3.18 processMessageAck . . . . .	1204
6.202.3.19 processMessageDispatch . . . . .	1204
6.202.3.20 processMessageDispatchNotification . . . . .	1204
6.202.3.21 processMessagePull . . . . .	1204
6.202.3.22 processPrepareTransaction . . . . .	1204
6.202.3.23 processProducerAck . . . . .	1205
6.202.3.24 processProducerInfo . . . . .	1205
6.202.3.25 processRecoverTransactions . . . . .	1205
6.202.3.26 processRemoveConnection . . . . .	1205
6.202.3.27 processRemoveConsumer . . . . .	1205
6.202.3.28 processRemoveDestination . . . . .	1205
6.202.3.29 processRemoveInfo . . . . .	1205
6.202.3.30 processRemoveProducer . . . . .	1205
6.202.3.31 processRemoveSession . . . . .	1206
6.202.3.32 processRemoveSubscriptionInfo . . . . .	1206
6.202.3.33 processReplayCommand . . . . .	1206
6.202.3.34 processResponse . . . . .	1206
6.202.3.35 processRollbackTransaction . . . . .	1206
6.202.3.36 processSessionInfo . . . . .	1206
6.202.3.37 processShutdownInfo . . . . .	1206
6.202.3.38 processTransactionInfo . . . . .	1206
6.202.3.39 processWireFormat . . . . .	1207
6.203 activemq::state::CommandVisitorAdapter Class Reference . . . . .	1208
6.203.1 Detailed Description . . . . .	1210
6.203.2 Constructor & Destructor Documentation . . . . .	1211
6.203.2.1 ~CommandVisitorAdapter . . . . .	1211
6.203.3 Member Function Documentation . . . . .	1211

6.203.3.1	processBeginTransaction	1211
6.203.3.2	processBrokerError	1211
6.203.3.3	processBrokerInfo	1211
6.203.3.4	processCommitTransactionOnePhase	1211
6.203.3.5	processCommitTransactionTwoPhase	1211
6.203.3.6	processConnectionControl	1211
6.203.3.7	processConnectionError	1211
6.203.3.8	processConnectionInfo	1211
6.203.3.9	processConsumerControl	1211
6.203.3.10	processConsumerInfo	1211
6.203.3.11	processControlCommand	1211
6.203.3.12	processDestinationInfo	1211
6.203.3.13	processEndTransaction	1211
6.203.3.14	processFlushCommand	1211
6.203.3.15	processForgetTransaction	1211
6.203.3.16	processKeepAliveInfo	1211
6.203.3.17	processMessage	1211
6.203.3.18	processMessageAck	1211
6.203.3.19	processMessageDispatch	1211
6.203.3.20	processMessageDispatchNotification	1211
6.203.3.21	processMessagePull	1211
6.203.3.22	processPrepareTransaction	1211
6.203.3.23	processProducerAck	1211
6.203.3.24	processProducerInfo	1211
6.203.3.25	processRecoverTransactions	1211
6.203.3.26	processRemoveConnection	1211
6.203.3.27	processRemoveConsumer	1211
6.203.3.28	processRemoveDestination	1211
6.203.3.29	processRemoveInfo	1211
6.203.3.30	processRemoveProducer	1212
6.203.3.31	processRemoveSession	1212
6.203.3.32	processRemoveSubscriptionInfo	1212
6.203.3.33	processReplayCommand	1212
6.203.3.34	processResponse	1212
6.203.3.35	processRollbackTransaction	1212
6.203.3.36	processSessionInfo	1212



6.203.3.37	processShutdownInfo . . . . .	1212
6.203.3.38	processTransactionInfo . . . . .	1212
6.203.3.39	processWireFormat . . . . .	1213
6.204	decaf::lang::Comparable< T > Class Template Reference . . . . .	1214
6.204.1	Detailed Description . . . . .	1214
6.204.2	Constructor & Destructor Documentation . . . . .	1214
6.204.2.1	~Comparable . . . . .	1214
6.204.3	Member Function Documentation . . . . .	1214
6.204.3.1	compareTo . . . . .	1214
6.204.3.2	equals . . . . .	1215
6.204.3.3	operator< . . . . .	1215
6.204.3.4	operator== . . . . .	1215
6.205	decaf::util::Comparator< T > Class Template Reference . . . . .	1217
6.205.1	Detailed Description . . . . .	1217
6.205.2	Constructor & Destructor Documentation . . . . .	1217
6.205.2.1	~Comparator . . . . .	1217
6.205.3	Member Function Documentation . . . . .	1217
6.205.3.1	compare . . . . .	1217
6.205.3.2	operator() . . . . .	1218
6.206	activemq::util::CompositeData Class Reference . . . . .	1219
6.206.1	Detailed Description . . . . .	1219
6.206.2	Constructor & Destructor Documentation . . . . .	1220
6.206.2.1	CompositeData . . . . .	1220
6.206.2.2	~CompositeData . . . . .	1220
6.206.3	Member Function Documentation . . . . .	1220
6.206.3.1	getComponents . . . . .	1220
6.206.3.2	getComponents . . . . .	1220
6.206.3.3	getFragment . . . . .	1220
6.206.3.4	getHost . . . . .	1220
6.206.3.5	getParameters . . . . .	1220
6.206.3.6	getPath . . . . .	1220
6.206.3.7	getScheme . . . . .	1220
6.206.3.8	setComponents . . . . .	1220
6.206.3.9	setFragment . . . . .	1220
6.206.3.10	setHost . . . . .	1220
6.206.3.11	setParameters . . . . .	1220

6.206.3.12	setPath . . . . .	1220
6.206.3.13	setScheme . . . . .	1220
6.206.3.14	oURI . . . . .	1220
6.207	activemq::threads::CompositeTask Class Reference . . . . .	1221
6.207.1	Detailed Description . . . . .	1221
6.207.2	Constructor & Destructor Documentation . . . . .	1221
6.207.2.1	~CompositeTask . . . . .	1221
6.207.3	Member Function Documentation . . . . .	1221
6.207.3.1	isPending . . . . .	1221
6.208	activemq::threads::CompositeTaskRunner Class Reference . . . . .	1223
6.208.1	Detailed Description . . . . .	1223
6.208.2	Constructor & Destructor Documentation . . . . .	1224
6.208.2.1	CompositeTaskRunner . . . . .	1224
6.208.2.2	~CompositeTaskRunner . . . . .	1224
6.208.3	Member Function Documentation . . . . .	1224
6.208.3.1	addTask . . . . .	1224
6.208.3.2	iterate . . . . .	1224
6.208.3.3	removeTask . . . . .	1224
6.208.3.4	run . . . . .	1224
6.208.3.5	shutdown . . . . .	1224
6.208.3.6	shutdown . . . . .	1225
6.208.3.7	wakeup . . . . .	1225
6.209	activemq::transport::CompositeTransport Class Reference . . . . .	1226
6.209.1	Detailed Description . . . . .	1226
6.209.2	Constructor & Destructor Documentation . . . . .	1226
6.209.2.1	~CompositeTransport . . . . .	1226
6.209.3	Member Function Documentation . . . . .	1226
6.209.3.1	addURI . . . . .	1226
6.209.3.2	removeURI . . . . .	1227
6.210	decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference . . . . .	1228
6.210.1	Detailed Description . . . . .	1228
6.210.2	Constructor & Destructor Documentation . . . . .	1229
6.210.2.1	~ConcurrentMap . . . . .	1229
6.210.3	Member Function Documentation . . . . .	1229
6.210.3.1	putIfAbsent . . . . .	1229
6.210.3.2	remove . . . . .	1230

6.210.3.3 replace . . . . .	1230
6.210.3.4 replace . . . . .	1231
6.211decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference . . . . .	1233
6.211.1 Detailed Description . . . . .	1236
6.211.2 Constructor & Destructor Documentation . . . . .	1237
6.211.2.1 ConcurrentStlMap . . . . .	1237
6.211.2.2 ConcurrentStlMap . . . . .	1237
6.211.2.3 ConcurrentStlMap . . . . .	1237
6.211.2.4 ~ConcurrentStlMap . . . . .	1237
6.211.3 Member Function Documentation . . . . .	1237
6.211.3.1 clear . . . . .	1237
6.211.3.2 containsKey . . . . .	1238
6.211.3.3 containsValue . . . . .	1238
6.211.3.4 copy . . . . .	1239
6.211.3.5 copy . . . . .	1239
6.211.3.6 equals . . . . .	1239
6.211.3.7 equals . . . . .	1239
6.211.3.8 get . . . . .	1239
6.211.3.9 get . . . . .	1240
6.211.3.10isEmpty . . . . .	1240
6.211.3.11keySet . . . . .	1240
6.211.3.12lock . . . . .	1241
6.211.3.13notify . . . . .	1241
6.211.3.14notifyAll . . . . .	1241
6.211.3.15put . . . . .	1242
6.211.3.16putAll . . . . .	1242
6.211.3.17putAll . . . . .	1242
6.211.3.18putIfAbsent . . . . .	1243
6.211.3.19remove . . . . .	1243
6.211.3.20remove . . . . .	1244
6.211.3.21replace . . . . .	1244
6.211.3.22replace . . . . .	1245
6.211.3.23size . . . . .	1246
6.211.3.24tryLock . . . . .	1246
6.211.3.25unlock . . . . .	1246
6.211.3.26values . . . . .	1246

6.211.3.27wait . . . . .	1247
6.211.3.28wait . . . . .	1247
6.211.3.29wait . . . . .	1248
6.212decaf::util::concurrent::locks::Condition Class Reference . . . . .	1249
6.212.1 Detailed Description . . . . .	1249
6.212.2 Constructor & Destructor Documentation . . . . .	1251
6.212.2.1 ~Condition . . . . .	1251
6.212.3 Member Function Documentation . . . . .	1251
6.212.3.1 await . . . . .	1251
6.212.3.2 await . . . . .	1251
6.212.3.3 awaitNanos . . . . .	1252
6.212.3.4 awaitUninterruptibly . . . . .	1253
6.212.3.5 awaitUntil . . . . .	1254
6.212.3.6 signal . . . . .	1254
6.212.3.7 signalAll . . . . .	1254
6.213decaf::util::concurrent::ConditionHandle Class Reference . . . . .	1255
6.213.1 Constructor & Destructor Documentation . . . . .	1255
6.213.1.1 ConditionHandle . . . . .	1255
6.213.1.2 ~ConditionHandle . . . . .	1255
6.213.1.3 ConditionHandle . . . . .	1255
6.213.1.4 ~ConditionHandle . . . . .	1255
6.213.2 Field Documentation . . . . .	1255
6.213.2.1 condition . . . . .	1255
6.213.2.2 criticalSection . . . . .	1255
6.213.2.3 generation . . . . .	1255
6.213.2.4 mutex . . . . .	1255
6.213.2.5 numWaiting . . . . .	1255
6.213.2.6 numWake . . . . .	1255
6.213.2.7 semaphore . . . . .	1255
6.214decaf::internal::util::concurrent::ConditionImpl Class Reference . . . . .	1257
6.214.1 Member Function Documentation . . . . .	1257
6.214.1.1 create . . . . .	1257
6.214.1.2 destroy . . . . .	1257
6.214.1.3 notify . . . . .	1258
6.214.1.4 notifyAll . . . . .	1258
6.214.1.5 wait . . . . .	1258

6.214.1.6 wait . . . . .	1258
6.215decaf::net::ConnectException Class Reference . . . . .	1259
6.215.1 Constructor & Destructor Documentation . . . . .	1259
6.215.1.1 ConnectException . . . . .	1259
6.215.1.2 ConnectException . . . . .	1259
6.215.1.3 ConnectException . . . . .	1260
6.215.1.4 ConnectException . . . . .	1260
6.215.1.5 ConnectException . . . . .	1260
6.215.1.6 ConnectException . . . . .	1260
6.215.1.7 ~ConnectException . . . . .	1261
6.215.2 Member Function Documentation . . . . .	1261
6.215.2.1 clone . . . . .	1261
6.216cms::Connection Class Reference . . . . .	1262
6.216.1 Detailed Description . . . . .	1262
6.216.2 Constructor & Destructor Documentation . . . . .	1263
6.216.2.1 ~Connection . . . . .	1263
6.216.3 Member Function Documentation . . . . .	1263
6.216.3.1 close . . . . .	1263
6.216.3.2 createSession . . . . .	1264
6.216.3.3 createSession . . . . .	1264
6.216.3.4 getClientID . . . . .	1264
6.216.3.5 getExceptionListener . . . . .	1264
6.216.3.6 getMetaData . . . . .	1265
6.216.3.7 setClientID . . . . .	1265
6.216.3.8 setExceptionListener . . . . .	1265
6.217activemq::commands::ConnectionControl Class Reference . . . . .	1267
6.217.1 Constructor & Destructor Documentation . . . . .	1268
6.217.1.1 ConnectionControl . . . . .	1268
6.217.1.2 ~ConnectionControl . . . . .	1268
6.217.2 Member Function Documentation . . . . .	1268
6.217.2.1 cloneDataStructure . . . . .	1268
6.217.2.2 copyDataStructure . . . . .	1268
6.217.2.3 equals . . . . .	1269
6.217.2.4 getConnectedBrokers . . . . .	1269
6.217.2.5 getConnectedBrokers . . . . .	1269
6.217.2.6 getDataStructureType . . . . .	1269

6.217.2.7	getReconnectTo	1270
6.217.2.8	getReconnectTo	1270
6.217.2.9	isClose	1270
6.217.2.10	isExit	1270
6.217.2.11	isFaultTolerant	1270
6.217.2.12	isRebalanceConnection	1270
6.217.2.13	isResume	1270
6.217.2.14	isSuspend	1270
6.217.2.15	setClose	1270
6.217.2.16	setConnectedBrokers	1270
6.217.2.17	setExit	1270
6.217.2.18	setFaultTolerant	1270
6.217.2.19	setRebalanceConnection	1270
6.217.2.20	setReconnectTo	1270
6.217.2.21	setResume	1270
6.217.2.22	setSuspend	1270
6.217.2.23	toString	1270
6.217.2.24	visit	1271
6.217.3	Field Documentation	1271
6.217.3.1	close	1271
6.217.3.2	connectedBrokers	1271
6.217.3.3	exit	1271
6.217.3.4	faultTolerant	1271
6.217.3.5	ID_CONNECTIONCONTROL	1271
6.217.3.6	rebalanceConnection	1271
6.217.3.7	reconnectTo	1271
6.217.3.8	resume	1271
6.217.3.9	suspend	1271
6.218	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class	
	Reference	1272
6.218.1	Detailed Description	1272
6.218.2	Constructor & Destructor Documentation	1273
6.218.2.1	ConnectionControlMarshaller	1273
6.218.2.2	~ConnectionControlMarshaller	1273
6.218.3	Member Function Documentation	1273
6.218.3.1	createObject	1273
6.218.3.2	getDataStructureType	1273

6.218.3.3 looseMarshal . . . . .	1273
6.218.3.4 looseUnmarshal . . . . .	1274
6.218.3.5 tightMarshal1 . . . . .	1274
6.218.3.6 tightMarshal2 . . . . .	1274
6.218.3.7 tightUnmarshal . . . . .	1275
6.219activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class	
Reference . . . . .	1276
6.219.1 Detailed Description . . . . .	1276
6.219.2 Constructor & Destructor Documentation . . . . .	1277
6.219.2.1 ConnectionControlMarshaller . . . . .	1277
6.219.2.2 ~ConnectionControlMarshaller . . . . .	1277
6.219.3 Member Function Documentation . . . . .	1277
6.219.3.1 createObject . . . . .	1277
6.219.3.2 getDataStructureType . . . . .	1277
6.219.3.3 looseMarshal . . . . .	1277
6.219.3.4 looseUnmarshal . . . . .	1278
6.219.3.5 tightMarshal1 . . . . .	1278
6.219.3.6 tightMarshal2 . . . . .	1278
6.219.3.7 tightUnmarshal . . . . .	1279
6.220activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class	
Reference . . . . .	1280
6.220.1 Detailed Description . . . . .	1280
6.220.2 Constructor & Destructor Documentation . . . . .	1281
6.220.2.1 ConnectionControlMarshaller . . . . .	1281
6.220.2.2 ~ConnectionControlMarshaller . . . . .	1281
6.220.3 Member Function Documentation . . . . .	1281
6.220.3.1 createObject . . . . .	1281
6.220.3.2 getDataStructureType . . . . .	1281
6.220.3.3 looseMarshal . . . . .	1281
6.220.3.4 looseUnmarshal . . . . .	1282
6.220.3.5 tightMarshal1 . . . . .	1282
6.220.3.6 tightMarshal2 . . . . .	1282
6.220.3.7 tightUnmarshal . . . . .	1283
6.221activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class	
Reference . . . . .	1284
6.221.1 Detailed Description . . . . .	1284
6.221.2 Constructor & Destructor Documentation . . . . .	1285

6.221.2.1	ConnectionControlMarshaller . . . . .	1285
6.221.2.2	~ConnectionControlMarshaller . . . . .	1285
6.221.3	Member Function Documentation . . . . .	1285
6.221.3.1	createObject . . . . .	1285
6.221.3.2	getDataStructureType . . . . .	1285
6.221.3.3	looseMarshal . . . . .	1285
6.221.3.4	looseUnmarshal . . . . .	1286
6.221.3.5	tightMarshal1 . . . . .	1286
6.221.3.6	tightMarshal2 . . . . .	1286
6.221.3.7	tightUnmarshal . . . . .	1287
6.222	activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller Class	
	Reference . . . . .	1288
6.222.1	Detailed Description . . . . .	1288
6.222.2	Constructor & Destructor Documentation . . . . .	1289
6.222.2.1	ConnectionControlMarshaller . . . . .	1289
6.222.2.2	~ConnectionControlMarshaller . . . . .	1289
6.222.3	Member Function Documentation . . . . .	1289
6.222.3.1	createObject . . . . .	1289
6.222.3.2	getDataStructureType . . . . .	1289
6.222.3.3	looseMarshal . . . . .	1289
6.222.3.4	looseUnmarshal . . . . .	1290
6.222.3.5	tightMarshal1 . . . . .	1290
6.222.3.6	tightMarshal2 . . . . .	1290
6.222.3.7	tightUnmarshal . . . . .	1291
6.223	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class	
	Reference . . . . .	1292
6.223.1	Detailed Description . . . . .	1292
6.223.2	Constructor & Destructor Documentation . . . . .	1293
6.223.2.1	ConnectionControlMarshaller . . . . .	1293
6.223.2.2	~ConnectionControlMarshaller . . . . .	1293
6.223.3	Member Function Documentation . . . . .	1293
6.223.3.1	createObject . . . . .	1293
6.223.3.2	getDataStructureType . . . . .	1293
6.223.3.3	looseMarshal . . . . .	1293
6.223.3.4	looseUnmarshal . . . . .	1294
6.223.3.5	tightMarshal1 . . . . .	1294
6.223.3.6	tightMarshal2 . . . . .	1294



6.223.3.7 tightUnmarshal . . . . .	1295
6.224activemq::commands::ConnectionError Class Reference . . . . .	1296
6.224.1 Constructor & Destructor Documentation . . . . .	1297
6.224.1.1 ConnectionError . . . . .	1297
6.224.1.2 ~ConnectionError . . . . .	1297
6.224.2 Member Function Documentation . . . . .	1297
6.224.2.1 cloneDataStructure . . . . .	1297
6.224.2.2 copyDataStructure . . . . .	1297
6.224.2.3 equals . . . . .	1297
6.224.2.4 getConnectionId . . . . .	1298
6.224.2.5 getConnectionId . . . . .	1298
6.224.2.6 getDataStructureType . . . . .	1298
6.224.2.7 getException . . . . .	1298
6.224.2.8 getException . . . . .	1298
6.224.2.9 setConnectionId . . . . .	1298
6.224.2.10 setException . . . . .	1298
6.224.2.11 toString . . . . .	1298
6.224.2.12 visit . . . . .	1298
6.224.3 Field Documentation . . . . .	1299
6.224.3.1 connectionId . . . . .	1299
6.224.3.2 exception . . . . .	1299
6.224.3.3 ID_CONNECTIONERROR . . . . .	1299
6.225activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference . . . . .	1300
6.225.1 Detailed Description . . . . .	1300
6.225.2 Constructor & Destructor Documentation . . . . .	1301
6.225.2.1 ConnectionErrorMarshaller . . . . .	1301
6.225.2.2 ~ConnectionErrorMarshaller . . . . .	1301
6.225.3 Member Function Documentation . . . . .	1301
6.225.3.1 createObject . . . . .	1301
6.225.3.2 getDataStructureType . . . . .	1301
6.225.3.3 looseMarshal . . . . .	1301
6.225.3.4 looseUnmarshal . . . . .	1302
6.225.3.5 tightMarshal1 . . . . .	1302
6.225.3.6 tightMarshal2 . . . . .	1302
6.225.3.7 tightUnmarshal . . . . .	1303

6.226	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	Class	
	Reference		1304
6.226.1	Detailed Description		1304
6.226.2	Constructor & Destructor Documentation		1305
6.226.2.1	ConnectionErrorMarshaller		1305
6.226.2.2	~ConnectionErrorMarshaller		1305
6.226.3	Member Function Documentation		1305
6.226.3.1	createObject		1305
6.226.3.2	getDataStructureType		1305
6.226.3.3	looseMarshal		1305
6.226.3.4	looseUnmarshal		1306
6.226.3.5	tightMarshal1		1306
6.226.3.6	tightMarshal2		1306
6.226.3.7	tightUnmarshal		1307
6.227	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	Class	
	Reference		1308
6.227.1	Detailed Description		1308
6.227.2	Constructor & Destructor Documentation		1309
6.227.2.1	ConnectionErrorMarshaller		1309
6.227.2.2	~ConnectionErrorMarshaller		1309
6.227.3	Member Function Documentation		1309
6.227.3.1	createObject		1309
6.227.3.2	getDataStructureType		1309
6.227.3.3	looseMarshal		1309
6.227.3.4	looseUnmarshal		1310
6.227.3.5	tightMarshal1		1310
6.227.3.6	tightMarshal2		1310
6.227.3.7	tightUnmarshal		1311
6.228	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	Class	
	Reference		1312
6.228.1	Detailed Description		1312
6.228.2	Constructor & Destructor Documentation		1313
6.228.2.1	ConnectionErrorMarshaller		1313
6.228.2.2	~ConnectionErrorMarshaller		1313
6.228.3	Member Function Documentation		1313
6.228.3.1	createObject		1313
6.228.3.2	getDataStructureType		1313

6.228.3.3 looseMarshal . . . . .	1313
6.228.3.4 looseUnmarshal . . . . .	1314
6.228.3.5 tightMarshal1 . . . . .	1314
6.228.3.6 tightMarshal2 . . . . .	1314
6.228.3.7 tightUnmarshal . . . . .	1315
6.229activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller Class	
Reference . . . . .	1316
6.229.1 Detailed Description . . . . .	1316
6.229.2 Constructor & Destructor Documentation . . . . .	1317
6.229.2.1 ConnectionErrorMarshaller . . . . .	1317
6.229.2.2 ~ConnectionErrorMarshaller . . . . .	1317
6.229.3 Member Function Documentation . . . . .	1317
6.229.3.1 createObject . . . . .	1317
6.229.3.2 getDataStructureType . . . . .	1317
6.229.3.3 looseMarshal . . . . .	1317
6.229.3.4 looseUnmarshal . . . . .	1318
6.229.3.5 tightMarshal1 . . . . .	1318
6.229.3.6 tightMarshal2 . . . . .	1318
6.229.3.7 tightUnmarshal . . . . .	1319
6.230activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller Class	
Reference . . . . .	1320
6.230.1 Detailed Description . . . . .	1320
6.230.2 Constructor & Destructor Documentation . . . . .	1321
6.230.2.1 ConnectionErrorMarshaller . . . . .	1321
6.230.2.2 ~ConnectionErrorMarshaller . . . . .	1321
6.230.3 Member Function Documentation . . . . .	1321
6.230.3.1 createObject . . . . .	1321
6.230.3.2 getDataStructureType . . . . .	1321
6.230.3.3 looseMarshal . . . . .	1321
6.230.3.4 looseUnmarshal . . . . .	1322
6.230.3.5 tightMarshal1 . . . . .	1322
6.230.3.6 tightMarshal2 . . . . .	1322
6.230.3.7 tightUnmarshal . . . . .	1323
6.231cms::ConnectionFactory Class Reference . . . . .	1324
6.231.1 Detailed Description . . . . .	1324
6.231.2 Constructor & Destructor Documentation . . . . .	1325
6.231.2.1 ~ConnectionFactory . . . . .	1325

6.231.3 Member Function Documentation . . . . .	1325
6.231.3.1 createCMSConnectionFactory . . . . .	1325
6.231.3.2 createConnection . . . . .	1325
6.231.3.3 createConnection . . . . .	1326
6.231.3.4 createConnection . . . . .	1326
6.232activemq::commands::ConnectionId Class Reference . . . . .	1327
6.232.1 Member Typedef Documentation . . . . .	1328
6.232.1.1 COMPARATOR . . . . .	1328
6.232.2 Constructor & Destructor Documentation . . . . .	1328
6.232.2.1 ConnectionId . . . . .	1328
6.232.2.2 ConnectionId . . . . .	1328
6.232.2.3 ConnectionId . . . . .	1328
6.232.2.4 ConnectionId . . . . .	1328
6.232.2.5 ConnectionId . . . . .	1328
6.232.2.6 ~ConnectionId . . . . .	1328
6.232.3 Member Function Documentation . . . . .	1328
6.232.3.1 cloneDataStructure . . . . .	1328
6.232.3.2 compareTo . . . . .	1328
6.232.3.3 copyDataStructure . . . . .	1328
6.232.3.4 equals . . . . .	1329
6.232.3.5 equals . . . . .	1329
6.232.3.6 getDataStructureType . . . . .	1329
6.232.3.7 getValue . . . . .	1329
6.232.3.8 getValue . . . . .	1329
6.232.3.9 operator< . . . . .	1329
6.232.3.10operator= . . . . .	1329
6.232.3.11operator== . . . . .	1329
6.232.3.12setValue . . . . .	1329
6.232.3.13toString . . . . .	1329
6.232.4 Field Documentation . . . . .	1330
6.232.4.1 ID_CONNECTIONID . . . . .	1330
6.232.4.2 value . . . . .	1330
6.233activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Refer- ence . . . . .	1331
6.233.1 Detailed Description . . . . .	1331
6.233.2 Constructor & Destructor Documentation . . . . .	1332
6.233.2.1 ConnectionIdMarshaller . . . . .	1332

6.233.2.2 ~ConnectionIdMarshaller . . . . .	1332
6.233.3 Member Function Documentation . . . . .	1332
6.233.3.1 createObject . . . . .	1332
6.233.3.2 getDataStructureType . . . . .	1332
6.233.3.3 looseMarshal . . . . .	1332
6.233.3.4 looseUnmarshal . . . . .	1333
6.233.3.5 tightMarshal1 . . . . .	1333
6.233.3.6 tightMarshal2 . . . . .	1333
6.233.3.7 tightUnmarshal . . . . .	1334
6.234activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference . . . . .	1335
6.234.1 Detailed Description . . . . .	1335
6.234.2 Constructor & Destructor Documentation . . . . .	1336
6.234.2.1 ConnectionIdMarshaller . . . . .	1336
6.234.2.2 ~ConnectionIdMarshaller . . . . .	1336
6.234.3 Member Function Documentation . . . . .	1336
6.234.3.1 createObject . . . . .	1336
6.234.3.2 getDataStructureType . . . . .	1336
6.234.3.3 looseMarshal . . . . .	1336
6.234.3.4 looseUnmarshal . . . . .	1337
6.234.3.5 tightMarshal1 . . . . .	1337
6.234.3.6 tightMarshal2 . . . . .	1337
6.234.3.7 tightUnmarshal . . . . .	1338
6.235activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference . . . . .	1339
6.235.1 Detailed Description . . . . .	1339
6.235.2 Constructor & Destructor Documentation . . . . .	1340
6.235.2.1 ConnectionIdMarshaller . . . . .	1340
6.235.2.2 ~ConnectionIdMarshaller . . . . .	1340
6.235.3 Member Function Documentation . . . . .	1340
6.235.3.1 createObject . . . . .	1340
6.235.3.2 getDataStructureType . . . . .	1340
6.235.3.3 looseMarshal . . . . .	1340
6.235.3.4 looseUnmarshal . . . . .	1341
6.235.3.5 tightMarshal1 . . . . .	1341
6.235.3.6 tightMarshal2 . . . . .	1341
6.235.3.7 tightUnmarshal . . . . .	1342

6.236activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference . . . . .	1343
6.236.1 Detailed Description . . . . .	1343
6.236.2 Constructor & Destructor Documentation . . . . .	1344
6.236.2.1 ConnectionIdMarshaller . . . . .	1344
6.236.2.2 ~ConnectionIdMarshaller . . . . .	1344
6.236.3 Member Function Documentation . . . . .	1344
6.236.3.1 createObject . . . . .	1344
6.236.3.2 getDataStructureType . . . . .	1344
6.236.3.3 looseMarshal . . . . .	1344
6.236.3.4 looseUnmarshal . . . . .	1345
6.236.3.5 tightMarshal1 . . . . .	1345
6.236.3.6 tightMarshal2 . . . . .	1345
6.236.3.7 tightUnmarshal . . . . .	1346
6.237activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference . . . . .	1347
6.237.1 Detailed Description . . . . .	1347
6.237.2 Constructor & Destructor Documentation . . . . .	1348
6.237.2.1 ConnectionIdMarshaller . . . . .	1348
6.237.2.2 ~ConnectionIdMarshaller . . . . .	1348
6.237.3 Member Function Documentation . . . . .	1348
6.237.3.1 createObject . . . . .	1348
6.237.3.2 getDataStructureType . . . . .	1348
6.237.3.3 looseMarshal . . . . .	1348
6.237.3.4 looseUnmarshal . . . . .	1349
6.237.3.5 tightMarshal1 . . . . .	1349
6.237.3.6 tightMarshal2 . . . . .	1349
6.237.3.7 tightUnmarshal . . . . .	1350
6.238activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference . . . . .	1351
6.238.1 Detailed Description . . . . .	1351
6.238.2 Constructor & Destructor Documentation . . . . .	1352
6.238.2.1 ConnectionIdMarshaller . . . . .	1352
6.238.2.2 ~ConnectionIdMarshaller . . . . .	1352
6.238.3 Member Function Documentation . . . . .	1352
6.238.3.1 createObject . . . . .	1352
6.238.3.2 getDataStructureType . . . . .	1352

6.238.3.3 looseMarshal . . . . .	1352
6.238.3.4 looseUnmarshal . . . . .	1353
6.238.3.5 tightMarshal1 . . . . .	1353
6.238.3.6 tightMarshal2 . . . . .	1353
6.238.3.7 tightUnmarshal . . . . .	1354
6.239activemq::commands::ConnectionInfo Class Reference . . . . .	1355
6.239.1 Constructor & Destructor Documentation . . . . .	1356
6.239.1.1 ConnectionInfo . . . . .	1356
6.239.1.2 ~ConnectionInfo . . . . .	1356
6.239.2 Member Function Documentation . . . . .	1356
6.239.2.1 cloneDataStructure . . . . .	1356
6.239.2.2 copyDataStructure . . . . .	1357
6.239.2.3 createRemoveCommand . . . . .	1357
6.239.2.4 equals . . . . .	1357
6.239.2.5 getBrokerPath . . . . .	1357
6.239.2.6 getBrokerPath . . . . .	1357
6.239.2.7 getClientId . . . . .	1357
6.239.2.8 getClientId . . . . .	1357
6.239.2.9 getConnectionId . . . . .	1357
6.239.2.10getConnectionId . . . . .	1357
6.239.2.11getDataStructureType . . . . .	1357
6.239.2.12getPassword . . . . .	1358
6.239.2.13getPassword . . . . .	1358
6.239.2.14getUserName . . . . .	1358
6.239.2.15getUserName . . . . .	1358
6.239.2.16sBrokerMasterConnector . . . . .	1358
6.239.2.17sClientMaster . . . . .	1358
6.239.2.18sConnectionInfo . . . . .	1358
6.239.2.19sFaultTolerant . . . . .	1359
6.239.2.20sManageable . . . . .	1359
6.239.2.21setBrokerMasterConnector . . . . .	1359
6.239.2.22setBrokerPath . . . . .	1359
6.239.2.23setClientId . . . . .	1359
6.239.2.24setClientMaster . . . . .	1359
6.239.2.25setConnectionId . . . . .	1359
6.239.2.26setFaultTolerant . . . . .	1359

6.239.2.27	setManageable . . . . .	1359
6.239.2.28	setPassword . . . . .	1359
6.239.2.29	setUserName . . . . .	1359
6.239.2.30	toString . . . . .	1359
6.239.2.31	visit . . . . .	1359
6.239.3	Field Documentation . . . . .	1360
6.239.3.1	brokerMasterConnector . . . . .	1360
6.239.3.2	brokerPath . . . . .	1360
6.239.3.3	clientId . . . . .	1360
6.239.3.4	clientMaster . . . . .	1360
6.239.3.5	connectionId . . . . .	1360
6.239.3.6	faultTolerant . . . . .	1360
6.239.3.7	ID_CONNECTIONINFO . . . . .	1360
6.239.3.8	manageable . . . . .	1360
6.239.3.9	password . . . . .	1360
6.239.3.10	userName . . . . .	1360
6.240	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference . . . . .	1361
6.240.1	Detailed Description . . . . .	1361
6.240.2	Constructor & Destructor Documentation . . . . .	1362
6.240.2.1	ConnectionInfoMarshaller . . . . .	1362
6.240.2.2	~ConnectionInfoMarshaller . . . . .	1362
6.240.3	Member Function Documentation . . . . .	1362
6.240.3.1	createObject . . . . .	1362
6.240.3.2	getDataStructureType . . . . .	1362
6.240.3.3	looseMarshal . . . . .	1362
6.240.3.4	looseUnmarshal . . . . .	1363
6.240.3.5	tightMarshal1 . . . . .	1363
6.240.3.6	tightMarshal2 . . . . .	1363
6.240.3.7	tightUnmarshal . . . . .	1364
6.241	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference . . . . .	1365
6.241.1	Detailed Description . . . . .	1365
6.241.2	Constructor & Destructor Documentation . . . . .	1366
6.241.2.1	ConnectionInfoMarshaller . . . . .	1366
6.241.2.2	~ConnectionInfoMarshaller . . . . .	1366
6.241.3	Member Function Documentation . . . . .	1366



6.241.3.1 createObject . . . . .	1366
6.241.3.2 getDataStructureType . . . . .	1366
6.241.3.3 looseMarshal . . . . .	1366
6.241.3.4 looseUnmarshal . . . . .	1367
6.241.3.5 tightMarshal1 . . . . .	1367
6.241.3.6 tightMarshal2 . . . . .	1367
6.241.3.7 tightUnmarshal . . . . .	1368
6.242activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference . . . . .	1369
6.242.1 Detailed Description . . . . .	1369
6.242.2 Constructor & Destructor Documentation . . . . .	1370
6.242.2.1 ConnectionInfoMarshaller . . . . .	1370
6.242.2.2 ~ConnectionInfoMarshaller . . . . .	1370
6.242.3 Member Function Documentation . . . . .	1370
6.242.3.1 createObject . . . . .	1370
6.242.3.2 getDataStructureType . . . . .	1370
6.242.3.3 looseMarshal . . . . .	1370
6.242.3.4 looseUnmarshal . . . . .	1371
6.242.3.5 tightMarshal1 . . . . .	1371
6.242.3.6 tightMarshal2 . . . . .	1371
6.242.3.7 tightUnmarshal . . . . .	1372
6.243activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference . . . . .	1373
6.243.1 Detailed Description . . . . .	1373
6.243.2 Constructor & Destructor Documentation . . . . .	1374
6.243.2.1 ConnectionInfoMarshaller . . . . .	1374
6.243.2.2 ~ConnectionInfoMarshaller . . . . .	1374
6.243.3 Member Function Documentation . . . . .	1374
6.243.3.1 createObject . . . . .	1374
6.243.3.2 getDataStructureType . . . . .	1374
6.243.3.3 looseMarshal . . . . .	1374
6.243.3.4 looseUnmarshal . . . . .	1375
6.243.3.5 tightMarshal1 . . . . .	1375
6.243.3.6 tightMarshal2 . . . . .	1375
6.243.3.7 tightUnmarshal . . . . .	1376
6.244activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference . . . . .	1377

6.244.1 Detailed Description . . . . .	1377
6.244.2 Constructor & Destructor Documentation . . . . .	1378
6.244.2.1 ConnectionInfoMarshaller . . . . .	1378
6.244.2.2 ~ConnectionInfoMarshaller . . . . .	1378
6.244.3 Member Function Documentation . . . . .	1378
6.244.3.1 createObject . . . . .	1378
6.244.3.2 getDataStructureType . . . . .	1378
6.244.3.3 looseMarshal . . . . .	1378
6.244.3.4 looseUnmarshal . . . . .	1379
6.244.3.5 tightMarshal1 . . . . .	1379
6.244.3.6 tightMarshal2 . . . . .	1379
6.244.3.7 tightUnmarshal . . . . .	1380
6.245activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference . . . . .	1381
6.245.1 Detailed Description . . . . .	1381
6.245.2 Constructor & Destructor Documentation . . . . .	1382
6.245.2.1 ConnectionInfoMarshaller . . . . .	1382
6.245.2.2 ~ConnectionInfoMarshaller . . . . .	1382
6.245.3 Member Function Documentation . . . . .	1382
6.245.3.1 createObject . . . . .	1382
6.245.3.2 getDataStructureType . . . . .	1382
6.245.3.3 looseMarshal . . . . .	1382
6.245.3.4 looseUnmarshal . . . . .	1383
6.245.3.5 tightMarshal1 . . . . .	1383
6.245.3.6 tightMarshal2 . . . . .	1383
6.245.3.7 tightUnmarshal . . . . .	1384
6.246cms::ConnectionMetaData Class Reference . . . . .	1385
6.246.1 Detailed Description . . . . .	1385
6.246.2 Constructor & Destructor Documentation . . . . .	1386
6.246.2.1 ~ConnectionMetaData . . . . .	1386
6.246.3 Member Function Documentation . . . . .	1386
6.246.3.1 getCMSMajorVersion . . . . .	1386
6.246.3.2 getCMSMinorVersion . . . . .	1386
6.246.3.3 getCMSProviderName . . . . .	1386
6.246.3.4 getCMSVersion . . . . .	1387
6.246.3.5 getCMSXPropertyNames . . . . .	1387
6.246.3.6 getProviderMajorVersion . . . . .	1387

6.246.3.7	getProviderMinorVersion . . . . .	1387
6.246.3.8	getProviderVersion . . . . .	1388
6.247	activemq::state::ConnectionState Class Reference . . . . .	1389
6.247.1	Constructor & Destructor Documentation . . . . .	1390
6.247.1.1	ConnectionState . . . . .	1390
6.247.1.2	~ConnectionState . . . . .	1390
6.247.2	Member Function Documentation . . . . .	1390
6.247.2.1	addSession . . . . .	1390
6.247.2.2	addTempDestination . . . . .	1390
6.247.2.3	addTransactionState . . . . .	1390
6.247.2.4	checkShutdown . . . . .	1390
6.247.2.5	getInfo . . . . .	1390
6.247.2.6	getRecoveringPullConsumers . . . . .	1390
6.247.2.7	getSessionState . . . . .	1390
6.247.2.8	getSessionStates . . . . .	1390
6.247.2.9	getTempDesinations . . . . .	1390
6.247.2.10	getTransactionState . . . . .	1390
6.247.2.11	getTransactionStates . . . . .	1391
6.247.2.12	sConnectionInterruptProcessingComplete . . . . .	1391
6.247.2.13	removeSession . . . . .	1391
6.247.2.14	removeTempDestination . . . . .	1391
6.247.2.15	removeTransactionState . . . . .	1391
6.247.2.16	reset . . . . .	1391
6.247.2.17	setConnectionInterruptProcessingComplete . . . . .	1391
6.247.2.18	shutdown . . . . .	1391
6.247.2.19	oString . . . . .	1391
6.248	activemq::state::ConnectionStateTracker Class Reference . . . . .	1392
6.248.1	Constructor & Destructor Documentation . . . . .	1394
6.248.1.1	ConnectionStateTracker . . . . .	1394
6.248.1.2	~ConnectionStateTracker . . . . .	1394
6.248.2	Member Function Documentation . . . . .	1394
6.248.2.1	connectionInterruptProcessingComplete . . . . .	1394
6.248.2.2	getMaxCacheSize . . . . .	1394
6.248.2.3	isRestoreConsumers . . . . .	1394
6.248.2.4	isRestoreProducers . . . . .	1394
6.248.2.5	isRestoreSessions . . . . .	1394

6.248.2.6	isRestoreTransaction . . . . .	1394
6.248.2.7	isTrackMessages . . . . .	1394
6.248.2.8	isTrackTransactionProducers . . . . .	1394
6.248.2.9	isTrackTransactions . . . . .	1394
6.248.2.10	processBeginTransaction . . . . .	1394
6.248.2.11	processCommitTransactionOnePhase . . . . .	1394
6.248.2.12	processCommitTransactionTwoPhase . . . . .	1395
6.248.2.13	processConnectionInfo . . . . .	1395
6.248.2.14	processConsumerInfo . . . . .	1395
6.248.2.15	processDestinationInfo . . . . .	1395
6.248.2.16	processEndTransaction . . . . .	1395
6.248.2.17	processMessage . . . . .	1395
6.248.2.18	processMessageAck . . . . .	1395
6.248.2.19	processPrepareTransaction . . . . .	1396
6.248.2.20	processProducerInfo . . . . .	1396
6.248.2.21	processRemoveConnection . . . . .	1396
6.248.2.22	processRemoveConsumer . . . . .	1396
6.248.2.23	processRemoveDestination . . . . .	1396
6.248.2.24	processRemoveProducer . . . . .	1396
6.248.2.25	processRemoveSession . . . . .	1396
6.248.2.26	processRollbackTransaction . . . . .	1397
6.248.2.27	processSessionInfo . . . . .	1397
6.248.2.28	restore . . . . .	1398
6.248.2.29	setMaxCacheSize . . . . .	1398
6.248.2.30	setRestoreConsumers . . . . .	1398
6.248.2.31	setRestoreProducers . . . . .	1398
6.248.2.32	setRestoreSessions . . . . .	1398
6.248.2.33	setRestoreTransaction . . . . .	1398
6.248.2.34	setTrackMessages . . . . .	1398
6.248.2.35	setTrackTransactionProducers . . . . .	1398
6.248.2.36	setTrackTransactions . . . . .	1398
6.248.2.37	track . . . . .	1398
6.248.2.38	rackBack . . . . .	1398
6.248.2.39	ransportInterrupted . . . . .	1398
6.248.3	Friends And Related Function Documentation . . . . .	1398
6.248.3.1	RemoveTransactionAction . . . . .	1398

6.249	decaf::util::logging::ConsoleHandler Class Reference . . . . .	1399
6.249.1	Detailed Description . . . . .	1399
6.249.2	Constructor & Destructor Documentation . . . . .	1399
6.249.2.1	ConsoleHandler . . . . .	1399
6.249.2.2	~ConsoleHandler . . . . .	1399
6.249.3	Member Function Documentation . . . . .	1399
6.249.3.1	close . . . . .	1399
6.249.3.2	publish . . . . .	1400
6.250	activemq::commands::ConsumerControl Class Reference . . . . .	1401
6.250.1	Constructor & Destructor Documentation . . . . .	1402
6.250.1.1	ConsumerControl . . . . .	1402
6.250.1.2	~ConsumerControl . . . . .	1402
6.250.2	Member Function Documentation . . . . .	1402
6.250.2.1	cloneDataStructure . . . . .	1402
6.250.2.2	copyDataStructure . . . . .	1402
6.250.2.3	equals . . . . .	1402
6.250.2.4	getConsumerId . . . . .	1403
6.250.2.5	getConsumerId . . . . .	1403
6.250.2.6	getDataStructureType . . . . .	1403
6.250.2.7	getDestination . . . . .	1404
6.250.2.8	getDestination . . . . .	1404
6.250.2.9	getPrefetch . . . . .	1404
6.250.2.10	isClose . . . . .	1404
6.250.2.11	isFlush . . . . .	1404
6.250.2.12	isStart . . . . .	1404
6.250.2.13	isStop . . . . .	1404
6.250.2.14	setClose . . . . .	1404
6.250.2.15	setConsumerId . . . . .	1404
6.250.2.16	setDestination . . . . .	1404
6.250.2.17	setFlush . . . . .	1404
6.250.2.18	setPrefetch . . . . .	1404
6.250.2.19	setStart . . . . .	1404
6.250.2.20	setStop . . . . .	1404
6.250.2.21	toString . . . . .	1404
6.250.2.22	visit . . . . .	1405
6.250.3	Field Documentation . . . . .	1405

6.250.3.1 close . . . . .	1405
6.250.3.2 consumerId . . . . .	1405
6.250.3.3 destination . . . . .	1405
6.250.3.4 flush . . . . .	1405
6.250.3.5 ID_CONSUMERCONTROL . . . . .	1405
6.250.3.6 prefetch . . . . .	1405
6.250.3.7 start . . . . .	1405
6.250.3.8 stop . . . . .	1405
6.251activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class	
Reference . . . . .	1406
6.251.1 Detailed Description . . . . .	1406
6.251.2 Constructor & Destructor Documentation . . . . .	1407
6.251.2.1 ConsumerControlMarshaller . . . . .	1407
6.251.2.2 ~ConsumerControlMarshaller . . . . .	1407
6.251.3 Member Function Documentation . . . . .	1407
6.251.3.1 createObject . . . . .	1407
6.251.3.2 getDataStructureType . . . . .	1407
6.251.3.3 looseMarshal . . . . .	1407
6.251.3.4 looseUnmarshal . . . . .	1408
6.251.3.5 tightMarshal1 . . . . .	1408
6.251.3.6 tightMarshal2 . . . . .	1408
6.251.3.7 tightUnmarshal . . . . .	1409
6.252activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class	
Reference . . . . .	1410
6.252.1 Detailed Description . . . . .	1410
6.252.2 Constructor & Destructor Documentation . . . . .	1411
6.252.2.1 ConsumerControlMarshaller . . . . .	1411
6.252.2.2 ~ConsumerControlMarshaller . . . . .	1411
6.252.3 Member Function Documentation . . . . .	1411
6.252.3.1 createObject . . . . .	1411
6.252.3.2 getDataStructureType . . . . .	1411
6.252.3.3 looseMarshal . . . . .	1411
6.252.3.4 looseUnmarshal . . . . .	1412
6.252.3.5 tightMarshal1 . . . . .	1412
6.252.3.6 tightMarshal2 . . . . .	1412
6.252.3.7 tightUnmarshal . . . . .	1413

6.253	activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	Class	
	Reference		1414
6.253.1	Detailed Description		1414
6.253.2	Constructor & Destructor Documentation		1415
6.253.2.1	ConsumerControlMarshaller		1415
6.253.2.2	~ConsumerControlMarshaller		1415
6.253.3	Member Function Documentation		1415
6.253.3.1	createObject		1415
6.253.3.2	getDataStructureType		1415
6.253.3.3	looseMarshal		1415
6.253.3.4	looseUnmarshal		1416
6.253.3.5	tightMarshal1		1416
6.253.3.6	tightMarshal2		1416
6.253.3.7	tightUnmarshal		1417
6.254	activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	Class	
	Reference		1418
6.254.1	Detailed Description		1418
6.254.2	Constructor & Destructor Documentation		1419
6.254.2.1	ConsumerControlMarshaller		1419
6.254.2.2	~ConsumerControlMarshaller		1419
6.254.3	Member Function Documentation		1419
6.254.3.1	createObject		1419
6.254.3.2	getDataStructureType		1419
6.254.3.3	looseMarshal		1419
6.254.3.4	looseUnmarshal		1420
6.254.3.5	tightMarshal1		1420
6.254.3.6	tightMarshal2		1420
6.254.3.7	tightUnmarshal		1421
6.255	activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	Class	
	Reference		1422
6.255.1	Detailed Description		1422
6.255.2	Constructor & Destructor Documentation		1423
6.255.2.1	ConsumerControlMarshaller		1423
6.255.2.2	~ConsumerControlMarshaller		1423
6.255.3	Member Function Documentation		1423
6.255.3.1	createObject		1423
6.255.3.2	getDataStructureType		1423

6.255.3.3 looseMarshal . . . . .	1423
6.255.3.4 looseUnmarshal . . . . .	1424
6.255.3.5 tightMarshal1 . . . . .	1424
6.255.3.6 tightMarshal2 . . . . .	1424
6.255.3.7 tightUnmarshal . . . . .	1425
6.256activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller Class Reference . . . . .	1426
6.256.1 Detailed Description . . . . .	1426
6.256.2 Constructor & Destructor Documentation . . . . .	1427
6.256.2.1 ConsumerControlMarshaller . . . . .	1427
6.256.2.2 ~ConsumerControlMarshaller . . . . .	1427
6.256.3 Member Function Documentation . . . . .	1427
6.256.3.1 createObject . . . . .	1427
6.256.3.2 getDataStructureType . . . . .	1427
6.256.3.3 looseMarshal . . . . .	1427
6.256.3.4 looseUnmarshal . . . . .	1428
6.256.3.5 tightMarshal1 . . . . .	1428
6.256.3.6 tightMarshal2 . . . . .	1428
6.256.3.7 tightUnmarshal . . . . .	1429
6.257activemq::commands::ConsumerId Class Reference . . . . .	1430
6.257.1 Member Typedef Documentation . . . . .	1431
6.257.1.1 COMPARATOR . . . . .	1431
6.257.2 Constructor & Destructor Documentation . . . . .	1431
6.257.2.1 ConsumerId . . . . .	1431
6.257.2.2 ConsumerId . . . . .	1431
6.257.2.3 ConsumerId . . . . .	1431
6.257.2.4 ~ConsumerId . . . . .	1431
6.257.3 Member Function Documentation . . . . .	1431
6.257.3.1 cloneDataStructure . . . . .	1431
6.257.3.2 compareTo . . . . .	1431
6.257.3.3 copyDataStructure . . . . .	1431
6.257.3.4 equals . . . . .	1432
6.257.3.5 equals . . . . .	1432
6.257.3.6 getConnectionId . . . . .	1432
6.257.3.7 getConnectionId . . . . .	1432
6.257.3.8 getDataStructureType . . . . .	1432
6.257.3.9 getParentId . . . . .	1433



6.257.3.10	getSessionId . . . . .	1433
6.257.3.11	getValue . . . . .	1433
6.257.3.12	operator< . . . . .	1433
6.257.3.13	operator= . . . . .	1433
6.257.3.14	operator== . . . . .	1433
6.257.3.15	setConnectionId . . . . .	1433
6.257.3.16	setSessionId . . . . .	1433
6.257.3.17	setValue . . . . .	1433
6.257.3.18	toString . . . . .	1433
6.257.4	Field Documentation . . . . .	1433
6.257.4.1	connectionId . . . . .	1433
6.257.4.2	ID_CONSUMERID . . . . .	1433
6.257.4.3	sessionId . . . . .	1434
6.257.4.4	value . . . . .	1434
6.258	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference	1435
6.258.1	Detailed Description . . . . .	1435
6.258.2	Constructor & Destructor Documentation . . . . .	1436
6.258.2.1	ConsumerIdMarshaller . . . . .	1436
6.258.2.2	~ConsumerIdMarshaller . . . . .	1436
6.258.3	Member Function Documentation . . . . .	1436
6.258.3.1	createObject . . . . .	1436
6.258.3.2	getDataStructureType . . . . .	1436
6.258.3.3	looseMarshal . . . . .	1436
6.258.3.4	looseUnmarshal . . . . .	1437
6.258.3.5	tightMarshal1 . . . . .	1437
6.258.3.6	tightMarshal2 . . . . .	1437
6.258.3.7	tightUnmarshal . . . . .	1438
6.259	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference	1439
6.259.1	Detailed Description . . . . .	1439
6.259.2	Constructor & Destructor Documentation . . . . .	1440
6.259.2.1	ConsumerIdMarshaller . . . . .	1440
6.259.2.2	~ConsumerIdMarshaller . . . . .	1440
6.259.3	Member Function Documentation . . . . .	1440
6.259.3.1	createObject . . . . .	1440
6.259.3.2	getDataStructureType . . . . .	1440
6.259.3.3	looseMarshal . . . . .	1440

6.259.3.4 looseUnmarshal . . . . .	1441
6.259.3.5 tightMarshal1 . . . . .	1441
6.259.3.6 tightMarshal2 . . . . .	1441
6.259.3.7 tightUnmarshal . . . . .	1442
6.260activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference	1443
6.260.1 Detailed Description . . . . .	1443
6.260.2 Constructor & Destructor Documentation . . . . .	1444
6.260.2.1 ConsumerIdMarshaller . . . . .	1444
6.260.2.2 ~ConsumerIdMarshaller . . . . .	1444
6.260.3 Member Function Documentation . . . . .	1444
6.260.3.1 createObject . . . . .	1444
6.260.3.2 getDataStructureType . . . . .	1444
6.260.3.3 looseMarshal . . . . .	1444
6.260.3.4 looseUnmarshal . . . . .	1445
6.260.3.5 tightMarshal1 . . . . .	1445
6.260.3.6 tightMarshal2 . . . . .	1445
6.260.3.7 tightUnmarshal . . . . .	1446
6.261activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference	1447
6.261.1 Detailed Description . . . . .	1447
6.261.2 Constructor & Destructor Documentation . . . . .	1448
6.261.2.1 ConsumerIdMarshaller . . . . .	1448
6.261.2.2 ~ConsumerIdMarshaller . . . . .	1448
6.261.3 Member Function Documentation . . . . .	1448
6.261.3.1 createObject . . . . .	1448
6.261.3.2 getDataStructureType . . . . .	1448
6.261.3.3 looseMarshal . . . . .	1448
6.261.3.4 looseUnmarshal . . . . .	1449
6.261.3.5 tightMarshal1 . . . . .	1449
6.261.3.6 tightMarshal2 . . . . .	1449
6.261.3.7 tightUnmarshal . . . . .	1450
6.262activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference	1451
6.262.1 Detailed Description . . . . .	1451
6.262.2 Constructor & Destructor Documentation . . . . .	1452
6.262.2.1 ConsumerIdMarshaller . . . . .	1452
6.262.2.2 ~ConsumerIdMarshaller . . . . .	1452
6.262.3 Member Function Documentation . . . . .	1452

6.262.3.1 createObject . . . . .	1452
6.262.3.2 getDataStructureType . . . . .	1452
6.262.3.3 looseMarshal . . . . .	1452
6.262.3.4 looseUnmarshal . . . . .	1453
6.262.3.5 tightMarshal1 . . . . .	1453
6.262.3.6 tightMarshal2 . . . . .	1453
6.262.3.7 tightUnmarshal . . . . .	1454
6.263activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class Reference	1455
6.263.1 Detailed Description . . . . .	1455
6.263.2 Constructor & Destructor Documentation . . . . .	1456
6.263.2.1 ConsumerIdMarshaller . . . . .	1456
6.263.2.2 ~ConsumerIdMarshaller . . . . .	1456
6.263.3 Member Function Documentation . . . . .	1456
6.263.3.1 createObject . . . . .	1456
6.263.3.2 getDataStructureType . . . . .	1456
6.263.3.3 looseMarshal . . . . .	1456
6.263.3.4 looseUnmarshal . . . . .	1457
6.263.3.5 tightMarshal1 . . . . .	1457
6.263.3.6 tightMarshal2 . . . . .	1457
6.263.3.7 tightUnmarshal . . . . .	1458
6.264activemq::commands::ConsumerInfo Class Reference . . . . .	1459
6.264.1 Constructor & Destructor Documentation . . . . .	1461
6.264.1.1 ConsumerInfo . . . . .	1461
6.264.1.2 ~ConsumerInfo . . . . .	1461
6.264.2 Member Function Documentation . . . . .	1461
6.264.2.1 cloneDataStructure . . . . .	1461
6.264.2.2 copyDataStructure . . . . .	1461
6.264.2.3 createRemoveCommand . . . . .	1461
6.264.2.4 equals . . . . .	1461
6.264.2.5 getAdditionalPredicate . . . . .	1462
6.264.2.6 getAdditionalPredicate . . . . .	1462
6.264.2.7 getBrokerPath . . . . .	1462
6.264.2.8 getBrokerPath . . . . .	1462
6.264.2.9 getConsumerId . . . . .	1462
6.264.2.10getConsumerId . . . . .	1462
6.264.2.11getDataStructureType . . . . .	1462

6.264.2.12	getDestination . . . . .	1463
6.264.2.13	getDestination . . . . .	1463
6.264.2.14	getMaximumPendingMessageLimit . . . . .	1463
6.264.2.15	getNetworkConsumerPath . . . . .	1463
6.264.2.16	getNetworkConsumerPath . . . . .	1463
6.264.2.17	getPrefetchSize . . . . .	1463
6.264.2.18	getPriority . . . . .	1463
6.264.2.19	getSelector . . . . .	1463
6.264.2.20	getSelector . . . . .	1463
6.264.2.21	getSubscriptionName . . . . .	1463
6.264.2.22	getSubscriptionName . . . . .	1463
6.264.2.23	sBrowser . . . . .	1463
6.264.2.24	sConsumerInfo . . . . .	1463
6.264.2.25	sDispatchAsync . . . . .	1464
6.264.2.26	sExclusive . . . . .	1464
6.264.2.27	sNetworkSubscription . . . . .	1464
6.264.2.28	sNoLocal . . . . .	1464
6.264.2.29	sNoRangeAcks . . . . .	1464
6.264.2.30	sOptimizedAcknowledge . . . . .	1464
6.264.2.31	isRetroactive . . . . .	1464
6.264.2.32	setAdditionalPredicate . . . . .	1464
6.264.2.33	setBrokerPath . . . . .	1464
6.264.2.34	setBrowser . . . . .	1464
6.264.2.35	setConsumerId . . . . .	1464
6.264.2.36	setDestination . . . . .	1464
6.264.2.37	setDispatchAsync . . . . .	1464
6.264.2.38	setExclusive . . . . .	1464
6.264.2.39	setMaximumPendingMessageLimit . . . . .	1464
6.264.2.40	setNetworkConsumerPath . . . . .	1464
6.264.2.41	setNetworkSubscription . . . . .	1464
6.264.2.42	setNoLocal . . . . .	1464
6.264.2.43	setNoRangeAcks . . . . .	1464
6.264.2.44	setOptimizedAcknowledge . . . . .	1464
6.264.2.45	setPrefetchSize . . . . .	1464
6.264.2.46	setPriority . . . . .	1464
6.264.2.47	setRetroactive . . . . .	1464

6.264.2.4	setSelector . . . . .	1464
6.264.2.4	setSubscriptionName . . . . .	1464
6.264.2.5	toString . . . . .	1464
6.264.2.5	visit . . . . .	1465
6.264.3	Field Documentation . . . . .	1466
6.264.3.1	additionalPredicate . . . . .	1466
6.264.3.2	brokerPath . . . . .	1466
6.264.3.3	browser . . . . .	1466
6.264.3.4	consumerId . . . . .	1466
6.264.3.5	destination . . . . .	1466
6.264.3.6	dispatchAsync . . . . .	1466
6.264.3.7	exclusive . . . . .	1466
6.264.3.8	ID_CONSUMERINFO . . . . .	1466
6.264.3.9	maximumPendingMessageLimit . . . . .	1466
6.264.3.10	networkConsumerPath . . . . .	1466
6.264.3.11	networkSubscription . . . . .	1466
6.264.3.12	noLocal . . . . .	1466
6.264.3.13	noRangeAcks . . . . .	1466
6.264.3.14	optimizedAcknowledge . . . . .	1466
6.264.3.15	prefetchSize . . . . .	1466
6.264.3.16	priority . . . . .	1466
6.264.3.17	retroactive . . . . .	1466
6.264.3.18	selector . . . . .	1466
6.264.3.19	subscriptionName . . . . .	1466
6.265	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference . . . . .	1468
6.265.1	Detailed Description . . . . .	1468
6.265.2	Constructor & Destructor Documentation . . . . .	1469
6.265.2.1	ConsumerInfoMarshaller . . . . .	1469
6.265.2.2	~ConsumerInfoMarshaller . . . . .	1469
6.265.3	Member Function Documentation . . . . .	1469
6.265.3.1	createObject . . . . .	1469
6.265.3.2	getDataStructureType . . . . .	1469
6.265.3.3	looseMarshal . . . . .	1469
6.265.3.4	looseUnmarshal . . . . .	1470
6.265.3.5	tightMarshal1 . . . . .	1470
6.265.3.6	tightMarshal2 . . . . .	1470

6.265.3.7 tightUnmarshal . . . . .	1471
6.266activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference . . . . .	1472
6.266.1 Detailed Description . . . . .	1472
6.266.2 Constructor & Destructor Documentation . . . . .	1473
6.266.2.1 ConsumerInfoMarshaller . . . . .	1473
6.266.2.2 ~ConsumerInfoMarshaller . . . . .	1473
6.266.3 Member Function Documentation . . . . .	1473
6.266.3.1 createObject . . . . .	1473
6.266.3.2 getDataStructureType . . . . .	1473
6.266.3.3 looseMarshal . . . . .	1473
6.266.3.4 looseUnmarshal . . . . .	1474
6.266.3.5 tightMarshal1 . . . . .	1474
6.266.3.6 tightMarshal2 . . . . .	1474
6.266.3.7 tightUnmarshal . . . . .	1475
6.267activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference . . . . .	1476
6.267.1 Detailed Description . . . . .	1476
6.267.2 Constructor & Destructor Documentation . . . . .	1477
6.267.2.1 ConsumerInfoMarshaller . . . . .	1477
6.267.2.2 ~ConsumerInfoMarshaller . . . . .	1477
6.267.3 Member Function Documentation . . . . .	1477
6.267.3.1 createObject . . . . .	1477
6.267.3.2 getDataStructureType . . . . .	1477
6.267.3.3 looseMarshal . . . . .	1477
6.267.3.4 looseUnmarshal . . . . .	1478
6.267.3.5 tightMarshal1 . . . . .	1478
6.267.3.6 tightMarshal2 . . . . .	1478
6.267.3.7 tightUnmarshal . . . . .	1479
6.268activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference . . . . .	1480
6.268.1 Detailed Description . . . . .	1480
6.268.2 Constructor & Destructor Documentation . . . . .	1481
6.268.2.1 ConsumerInfoMarshaller . . . . .	1481
6.268.2.2 ~ConsumerInfoMarshaller . . . . .	1481
6.268.3 Member Function Documentation . . . . .	1481
6.268.3.1 createObject . . . . .	1481

6.268.3.2	getDataStructureType . . . . .	1481
6.268.3.3	looseMarshal . . . . .	1481
6.268.3.4	looseUnmarshal . . . . .	1482
6.268.3.5	tightMarshal1 . . . . .	1482
6.268.3.6	tightMarshal2 . . . . .	1482
6.268.3.7	tightUnmarshal . . . . .	1483
6.269	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference . . . . .	1484
6.269.1	Detailed Description . . . . .	1484
6.269.2	Constructor & Destructor Documentation . . . . .	1485
6.269.2.1	ConsumerInfoMarshaller . . . . .	1485
6.269.2.2	~ConsumerInfoMarshaller . . . . .	1485
6.269.3	Member Function Documentation . . . . .	1485
6.269.3.1	createObject . . . . .	1485
6.269.3.2	getDataStructureType . . . . .	1485
6.269.3.3	looseMarshal . . . . .	1485
6.269.3.4	looseUnmarshal . . . . .	1486
6.269.3.5	tightMarshal1 . . . . .	1486
6.269.3.6	tightMarshal2 . . . . .	1486
6.269.3.7	tightUnmarshal . . . . .	1487
6.270	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller Class Reference . . . . .	1488
6.270.1	Detailed Description . . . . .	1488
6.270.2	Constructor & Destructor Documentation . . . . .	1489
6.270.2.1	ConsumerInfoMarshaller . . . . .	1489
6.270.2.2	~ConsumerInfoMarshaller . . . . .	1489
6.270.3	Member Function Documentation . . . . .	1489
6.270.3.1	createObject . . . . .	1489
6.270.3.2	getDataStructureType . . . . .	1489
6.270.3.3	looseMarshal . . . . .	1489
6.270.3.4	looseUnmarshal . . . . .	1490
6.270.3.5	tightMarshal1 . . . . .	1490
6.270.3.6	tightMarshal2 . . . . .	1490
6.270.3.7	tightUnmarshal . . . . .	1491
6.271	activemq::state::ConsumerState Class Reference . . . . .	1492
6.271.1	Constructor & Destructor Documentation . . . . .	1492
6.271.1.1	ConsumerState . . . . .	1492

6.271.1.2 <code>~ConsumerState</code> . . . . .	1492
6.271.2 Member Function Documentation . . . . .	1492
6.271.2.1 <code>getInfo</code> . . . . .	1492
6.271.2.2 <code>toString</code> . . . . .	1492
6.272 <code>activemq::commands::ControlCommand</code> Class Reference . . . . .	1493
6.272.1 Constructor & Destructor Documentation . . . . .	1494
6.272.1.1 <code>ControlCommand</code> . . . . .	1494
6.272.1.2 <code>~ControlCommand</code> . . . . .	1494
6.272.2 Member Function Documentation . . . . .	1494
6.272.2.1 <code>cloneDataStructure</code> . . . . .	1494
6.272.2.2 <code>copyDataStructure</code> . . . . .	1494
6.272.2.3 <code>equals</code> . . . . .	1494
6.272.2.4 <code>getCommand</code> . . . . .	1495
6.272.2.5 <code>getCommand</code> . . . . .	1495
6.272.2.6 <code>getDataStructureType</code> . . . . .	1495
6.272.2.7 <code>setCommand</code> . . . . .	1495
6.272.2.8 <code>toString</code> . . . . .	1495
6.272.2.9 <code>visit</code> . . . . .	1495
6.272.3 Field Documentation . . . . .	1496
6.272.3.1 <code>command</code> . . . . .	1496
6.272.3.2 <code>ID_CONTROLCOMMAND</code> . . . . .	1496
6.273 <code>activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller</code> Class Reference . . . . .	1497
6.273.1 Detailed Description . . . . .	1497
6.273.2 Constructor & Destructor Documentation . . . . .	1498
6.273.2.1 <code>ControlCommandMarshaller</code> . . . . .	1498
6.273.2.2 <code>~ControlCommandMarshaller</code> . . . . .	1498
6.273.3 Member Function Documentation . . . . .	1498
6.273.3.1 <code>createObject</code> . . . . .	1498
6.273.3.2 <code>getDataStructureType</code> . . . . .	1498
6.273.3.3 <code>looseMarshal</code> . . . . .	1498
6.273.3.4 <code>looseUnmarshal</code> . . . . .	1499
6.273.3.5 <code>tightMarshal1</code> . . . . .	1499
6.273.3.6 <code>tightMarshal2</code> . . . . .	1499
6.273.3.7 <code>tightUnmarshal</code> . . . . .	1500
6.274 <code>activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller</code> Class Reference . . . . .	1501



6.274.1 Detailed Description . . . . .	1501
6.274.2 Constructor & Destructor Documentation . . . . .	1502
6.274.2.1 ControlCommandMarshaller . . . . .	1502
6.274.2.2 ~ControlCommandMarshaller . . . . .	1502
6.274.3 Member Function Documentation . . . . .	1502
6.274.3.1 createObject . . . . .	1502
6.274.3.2 getDataStructureType . . . . .	1502
6.274.3.3 looseMarshal . . . . .	1502
6.274.3.4 looseUnmarshal . . . . .	1503
6.274.3.5 tightMarshal1 . . . . .	1503
6.274.3.6 tightMarshal2 . . . . .	1503
6.274.3.7 tightUnmarshal . . . . .	1504
6.275activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class	
Reference . . . . .	1505
6.275.1 Detailed Description . . . . .	1505
6.275.2 Constructor & Destructor Documentation . . . . .	1506
6.275.2.1 ControlCommandMarshaller . . . . .	1506
6.275.2.2 ~ControlCommandMarshaller . . . . .	1506
6.275.3 Member Function Documentation . . . . .	1506
6.275.3.1 createObject . . . . .	1506
6.275.3.2 getDataStructureType . . . . .	1506
6.275.3.3 looseMarshal . . . . .	1506
6.275.3.4 looseUnmarshal . . . . .	1507
6.275.3.5 tightMarshal1 . . . . .	1507
6.275.3.6 tightMarshal2 . . . . .	1507
6.275.3.7 tightUnmarshal . . . . .	1508
6.276activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class	
Reference . . . . .	1509
6.276.1 Detailed Description . . . . .	1509
6.276.2 Constructor & Destructor Documentation . . . . .	1510
6.276.2.1 ControlCommandMarshaller . . . . .	1510
6.276.2.2 ~ControlCommandMarshaller . . . . .	1510
6.276.3 Member Function Documentation . . . . .	1510
6.276.3.1 createObject . . . . .	1510
6.276.3.2 getDataStructureType . . . . .	1510
6.276.3.3 looseMarshal . . . . .	1510
6.276.3.4 looseUnmarshal . . . . .	1511

6.276.3.5 tightMarshal1 . . . . .	1511
6.276.3.6 tightMarshal2 . . . . .	1511
6.276.3.7 tightUnmarshal . . . . .	1512
6.277activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class	
Reference . . . . .	1513
6.277.1 Detailed Description . . . . .	1513
6.277.2 Constructor & Destructor Documentation . . . . .	1514
6.277.2.1 ControlCommandMarshaller . . . . .	1514
6.277.2.2 ~ControlCommandMarshaller . . . . .	1514
6.277.3 Member Function Documentation . . . . .	1514
6.277.3.1 createObject . . . . .	1514
6.277.3.2 getDataStructureType . . . . .	1514
6.277.3.3 looseMarshal . . . . .	1514
6.277.3.4 looseUnmarshal . . . . .	1515
6.277.3.5 tightMarshal1 . . . . .	1515
6.277.3.6 tightMarshal2 . . . . .	1515
6.277.3.7 tightUnmarshal . . . . .	1516
6.278activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class	
Reference . . . . .	1517
6.278.1 Detailed Description . . . . .	1517
6.278.2 Constructor & Destructor Documentation . . . . .	1518
6.278.2.1 ControlCommandMarshaller . . . . .	1518
6.278.2.2 ~ControlCommandMarshaller . . . . .	1518
6.278.3 Member Function Documentation . . . . .	1518
6.278.3.1 createObject . . . . .	1518
6.278.3.2 getDataStructureType . . . . .	1518
6.278.3.3 looseMarshal . . . . .	1518
6.278.3.4 looseUnmarshal . . . . .	1519
6.278.3.5 tightMarshal1 . . . . .	1519
6.278.3.6 tightMarshal2 . . . . .	1519
6.278.3.7 tightUnmarshal . . . . .	1520
6.279decaf::util::concurrent::CountDownLatch Class Reference . . . . .	1521
6.279.1 Constructor & Destructor Documentation . . . . .	1521
6.279.1.1 CountDownLatch . . . . .	1521
6.279.1.2 ~CountDownLatch . . . . .	1522
6.279.2 Member Function Documentation . . . . .	1522
6.279.2.1 await . . . . .	1522

6.279.2.2	await	1522
6.279.2.3	await	1523
6.279.2.4	countDown	1523
6.279.2.5	getCount	1523
6.280	decaf::util::zip::CRC32 Class Reference	1524
6.280.1	Detailed Description	1524
6.280.2	Constructor & Destructor Documentation	1525
6.280.2.1	CRC32	1525
6.280.2.2	~CRC32	1525
6.280.3	Member Function Documentation	1525
6.280.3.1	getValue	1525
6.280.3.2	reset	1525
6.280.3.3	update	1525
6.280.3.4	update	1525
6.280.3.5	update	1526
6.280.3.6	update	1526
6.281	ct_data_s Struct Reference	1527
6.281.1	Field Documentation	1527
6.281.1.1	code	1527
6.281.1.2	dad	1527
6.281.1.3	dl	1527
6.281.1.4	fc	1527
6.281.1.5	freq	1527
6.281.1.6	len	1527
6.282	activemq::commands::DataArrayResponse Class Reference	1528
6.282.1	Constructor & Destructor Documentation	1529
6.282.1.1	DataArrayResponse	1529
6.282.1.2	~DataArrayResponse	1529
6.282.2	Member Function Documentation	1529
6.282.2.1	cloneDataStructure	1529
6.282.2.2	copyDataStructure	1529
6.282.2.3	equals	1529
6.282.2.4	getData	1530
6.282.2.5	getData	1530
6.282.2.6	getDataStructureType	1530
6.282.2.7	setData	1530

6.282.2.8 toString . . . . .	1530
6.282.3 Field Documentation . . . . .	1530
6.282.3.1 data . . . . .	1530
6.282.3.2 ID_DATAARRAYRESPONSE . . . . .	1530
6.283activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class	
Reference . . . . .	1531
6.283.1 Detailed Description . . . . .	1531
6.283.2 Constructor & Destructor Documentation . . . . .	1532
6.283.2.1 DataArrayResponseMarshaller . . . . .	1532
6.283.2.2 ~DataArrayResponseMarshaller . . . . .	1532
6.283.3 Member Function Documentation . . . . .	1532
6.283.3.1 createObject . . . . .	1532
6.283.3.2 getDataStructureType . . . . .	1532
6.283.3.3 looseMarshal . . . . .	1532
6.283.3.4 looseUnmarshal . . . . .	1533
6.283.3.5 tightMarshal1 . . . . .	1533
6.283.3.6 tightMarshal2 . . . . .	1534
6.283.3.7 tightUnmarshal . . . . .	1534
6.284activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class	
Reference . . . . .	1535
6.284.1 Detailed Description . . . . .	1535
6.284.2 Constructor & Destructor Documentation . . . . .	1536
6.284.2.1 DataArrayResponseMarshaller . . . . .	1536
6.284.2.2 ~DataArrayResponseMarshaller . . . . .	1536
6.284.3 Member Function Documentation . . . . .	1536
6.284.3.1 createObject . . . . .	1536
6.284.3.2 getDataStructureType . . . . .	1536
6.284.3.3 looseMarshal . . . . .	1536
6.284.3.4 looseUnmarshal . . . . .	1537
6.284.3.5 tightMarshal1 . . . . .	1537
6.284.3.6 tightMarshal2 . . . . .	1538
6.284.3.7 tightUnmarshal . . . . .	1538
6.285activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class	
Reference . . . . .	1539
6.285.1 Detailed Description . . . . .	1539
6.285.2 Constructor & Destructor Documentation . . . . .	1540
6.285.2.1 DataArrayResponseMarshaller . . . . .	1540

6.285.2.2	~DataArrayResponseMarshaller	1540
6.285.3	Member Function Documentation	1540
6.285.3.1	createObject	1540
6.285.3.2	getDataStructureType	1540
6.285.3.3	looseMarshal	1540
6.285.3.4	looseUnmarshal	1541
6.285.3.5	tightMarshal1	1541
6.285.3.6	tightMarshal2	1542
6.285.3.7	tightUnmarshal	1542
6.286	activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class	
	Reference	1543
6.286.1	Detailed Description	1543
6.286.2	Constructor & Destructor Documentation	1544
6.286.2.1	DataArrayResponseMarshaller	1544
6.286.2.2	~DataArrayResponseMarshaller	1544
6.286.3	Member Function Documentation	1544
6.286.3.1	createObject	1544
6.286.3.2	getDataStructureType	1544
6.286.3.3	looseMarshal	1544
6.286.3.4	looseUnmarshal	1545
6.286.3.5	tightMarshal1	1545
6.286.3.6	tightMarshal2	1546
6.286.3.7	tightUnmarshal	1546
6.287	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class	
	Reference	1547
6.287.1	Detailed Description	1547
6.287.2	Constructor & Destructor Documentation	1548
6.287.2.1	DataArrayResponseMarshaller	1548
6.287.2.2	~DataArrayResponseMarshaller	1548
6.287.3	Member Function Documentation	1548
6.287.3.1	createObject	1548
6.287.3.2	getDataStructureType	1548
6.287.3.3	looseMarshal	1548
6.287.3.4	looseUnmarshal	1549
6.287.3.5	tightMarshal1	1549
6.287.3.6	tightMarshal2	1550
6.287.3.7	tightUnmarshal	1550

6.288	activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference . . . . .	1551
6.288.1	Detailed Description . . . . .	1551
6.288.2	Constructor & Destructor Documentation . . . . .	1552
6.288.2.1	DataArrayResponseMarshaller . . . . .	1552
6.288.2.2	~DataArrayResponseMarshaller . . . . .	1552
6.288.3	Member Function Documentation . . . . .	1552
6.288.3.1	createObject . . . . .	1552
6.288.3.2	getDataStructureType . . . . .	1552
6.288.3.3	looseMarshal . . . . .	1552
6.288.3.4	looseUnmarshal . . . . .	1553
6.288.3.5	tightMarshal1 . . . . .	1553
6.288.3.6	tightMarshal2 . . . . .	1554
6.288.3.7	tightUnmarshal . . . . .	1554
6.289	decaf::util::zip::DataFormatException Class Reference . . . . .	1555
6.289.1	Constructor & Destructor Documentation . . . . .	1555
6.289.1.1	DataFormatException . . . . .	1555
6.289.1.2	DataFormatException . . . . .	1555
6.289.1.3	DataFormatException . . . . .	1556
6.289.1.4	DataFormatException . . . . .	1556
6.289.1.5	DataFormatException . . . . .	1556
6.289.1.6	DataFormatException . . . . .	1556
6.289.1.7	~DataFormatException . . . . .	1557
6.289.2	Member Function Documentation . . . . .	1557
6.289.2.1	clone . . . . .	1557
6.290	decaf::io::DataInput Class Reference . . . . .	1558
6.290.1	Detailed Description . . . . .	1559
6.290.2	Constructor & Destructor Documentation . . . . .	1559
6.290.2.1	~DataInput . . . . .	1559
6.290.3	Member Function Documentation . . . . .	1559
6.290.3.1	readBoolean . . . . .	1559
6.290.3.2	readByte . . . . .	1560
6.290.3.3	readChar . . . . .	1560
6.290.3.4	readDouble . . . . .	1560
6.290.3.5	readFloat . . . . .	1561
6.290.3.6	readFully . . . . .	1561
6.290.3.7	readFully . . . . .	1561

6.290.3.8 readInt . . . . .	1562
6.290.3.9 readLine . . . . .	1562
6.290.3.10 readLong . . . . .	1563
6.290.3.11 readShort . . . . .	1563
6.290.3.12 readString . . . . .	1563
6.290.3.13 readUnsignedByte . . . . .	1564
6.290.3.14 readUnsignedShort . . . . .	1564
6.290.3.15 readUTF . . . . .	1564
6.290.3.16 skipBytes . . . . .	1565
6.291 decaf::io::DataInputStream Class Reference . . . . .	1566
6.291.1 Detailed Description . . . . .	1567
6.291.2 Constructor & Destructor Documentation . . . . .	1567
6.291.2.1 DataInputStream . . . . .	1567
6.291.2.2 ~DataInputStream . . . . .	1568
6.291.3 Member Function Documentation . . . . .	1568
6.291.3.1 readBoolean . . . . .	1568
6.291.3.2 readByte . . . . .	1568
6.291.3.3 readChar . . . . .	1568
6.291.3.4 readDouble . . . . .	1569
6.291.3.5 readFloat . . . . .	1569
6.291.3.6 readFully . . . . .	1569
6.291.3.7 readFully . . . . .	1570
6.291.3.8 readInt . . . . .	1570
6.291.3.9 readLine . . . . .	1571
6.291.3.10 readLong . . . . .	1571
6.291.3.11 readShort . . . . .	1571
6.291.3.12 readString . . . . .	1572
6.291.3.13 readUnsignedByte . . . . .	1572
6.291.3.14 readUnsignedShort . . . . .	1572
6.291.3.15 readUTF . . . . .	1573
6.291.3.16 skipBytes . . . . .	1573
6.292 decaf::io::DataOutput Class Reference . . . . .	1574
6.292.1 Detailed Description . . . . .	1575
6.292.2 Constructor & Destructor Documentation . . . . .	1575
6.292.2.1 ~DataOutput . . . . .	1575
6.292.3 Member Function Documentation . . . . .	1575

6.292.3.1 writeBoolean . . . . .	1575
6.292.3.2 writeByte . . . . .	1575
6.292.3.3 writeBytes . . . . .	1576
6.292.3.4 writeChar . . . . .	1576
6.292.3.5 writeChars . . . . .	1576
6.292.3.6 writeDouble . . . . .	1576
6.292.3.7 writeFloat . . . . .	1577
6.292.3.8 writeInt . . . . .	1577
6.292.3.9 writeLong . . . . .	1577
6.292.3.10 writeShort . . . . .	1578
6.292.3.11 writeUnsignedShort . . . . .	1578
6.292.3.12 writeUTF . . . . .	1578
6.293 decaf::io::DataOutputStream Class Reference . . . . .	1579
6.293.1 Detailed Description . . . . .	1580
6.293.2 Constructor & Destructor Documentation . . . . .	1580
6.293.2.1 DataOutputStream . . . . .	1580
6.293.2.2 ~DataOutputStream . . . . .	1580
6.293.3 Member Function Documentation . . . . .	1580
6.293.3.1 doWriteArrayBounded . . . . .	1580
6.293.3.2 doWriteByte . . . . .	1581
6.293.3.3 size . . . . .	1581
6.293.3.4 writeBoolean . . . . .	1582
6.293.3.5 writeByte . . . . .	1582
6.293.3.6 writeBytes . . . . .	1582
6.293.3.7 writeChar . . . . .	1582
6.293.3.8 writeChars . . . . .	1582
6.293.3.9 writeDouble . . . . .	1582
6.293.3.10 writeFloat . . . . .	1582
6.293.3.11 writeInt . . . . .	1582
6.293.3.12 writeLong . . . . .	1582
6.293.3.13 writeShort . . . . .	1582
6.293.3.14 writeUnsignedShort . . . . .	1582
6.293.3.15 writeUTF . . . . .	1582
6.293.4 Field Documentation . . . . .	1582
6.293.4.1 buffer . . . . .	1582
6.293.4.2 written . . . . .	1582



6.294activemq::commands::DataResponse Class Reference . . . . .	1583
6.294.1 Constructor & Destructor Documentation . . . . .	1584
6.294.1.1 DataResponse . . . . .	1584
6.294.1.2 ~DataResponse . . . . .	1584
6.294.2 Member Function Documentation . . . . .	1584
6.294.2.1 cloneDataStructure . . . . .	1584
6.294.2.2 copyDataStructure . . . . .	1584
6.294.2.3 equals . . . . .	1584
6.294.2.4 getData . . . . .	1585
6.294.2.5 getData . . . . .	1585
6.294.2.6 getDataStructureType . . . . .	1585
6.294.2.7 setData . . . . .	1585
6.294.2.8 toString . . . . .	1585
6.294.3 Field Documentation . . . . .	1585
6.294.3.1 data . . . . .	1585
6.294.3.2 ID_DATARESPONSE . . . . .	1585
6.295activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Refer- ence . . . . .	1586
6.295.1 Detailed Description . . . . .	1586
6.295.2 Constructor & Destructor Documentation . . . . .	1587
6.295.2.1 DataResponseMarshaller . . . . .	1587
6.295.2.2 ~DataResponseMarshaller . . . . .	1587
6.295.3 Member Function Documentation . . . . .	1587
6.295.3.1 createObject . . . . .	1587
6.295.3.2 getDataStructureType . . . . .	1587
6.295.3.3 looseMarshal . . . . .	1587
6.295.3.4 looseUnmarshal . . . . .	1588
6.295.3.5 tightMarshal1 . . . . .	1588
6.295.3.6 tightMarshal2 . . . . .	1589
6.295.3.7 tightUnmarshal . . . . .	1589
6.296activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller Class Refer- ence . . . . .	1590
6.296.1 Detailed Description . . . . .	1590
6.296.2 Constructor & Destructor Documentation . . . . .	1591
6.296.2.1 DataResponseMarshaller . . . . .	1591
6.296.2.2 ~DataResponseMarshaller . . . . .	1591
6.296.3 Member Function Documentation . . . . .	1591

6.296.3.1 createObject . . . . .	1591
6.296.3.2 getDataStructureType . . . . .	1591
6.296.3.3 looseMarshal . . . . .	1591
6.296.3.4 looseUnmarshal . . . . .	1592
6.296.3.5 tightMarshal1 . . . . .	1592
6.296.3.6 tightMarshal2 . . . . .	1593
6.296.3.7 tightUnmarshal . . . . .	1593
6.297activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference . . . . .	1594
6.297.1 Detailed Description . . . . .	1594
6.297.2 Constructor & Destructor Documentation . . . . .	1595
6.297.2.1 DataResponseMarshaller . . . . .	1595
6.297.2.2 ~DataResponseMarshaller . . . . .	1595
6.297.3 Member Function Documentation . . . . .	1595
6.297.3.1 createObject . . . . .	1595
6.297.3.2 getDataStructureType . . . . .	1595
6.297.3.3 looseMarshal . . . . .	1595
6.297.3.4 looseUnmarshal . . . . .	1596
6.297.3.5 tightMarshal1 . . . . .	1596
6.297.3.6 tightMarshal2 . . . . .	1597
6.297.3.7 tightUnmarshal . . . . .	1597
6.298activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference . . . . .	1598
6.298.1 Detailed Description . . . . .	1598
6.298.2 Constructor & Destructor Documentation . . . . .	1599
6.298.2.1 DataResponseMarshaller . . . . .	1599
6.298.2.2 ~DataResponseMarshaller . . . . .	1599
6.298.3 Member Function Documentation . . . . .	1599
6.298.3.1 createObject . . . . .	1599
6.298.3.2 getDataStructureType . . . . .	1599
6.298.3.3 looseMarshal . . . . .	1599
6.298.3.4 looseUnmarshal . . . . .	1600
6.298.3.5 tightMarshal1 . . . . .	1600
6.298.3.6 tightMarshal2 . . . . .	1601
6.298.3.7 tightUnmarshal . . . . .	1601
6.299activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference . . . . .	1602

6.299.1 Detailed Description . . . . .	1602
6.299.2 Constructor & Destructor Documentation . . . . .	1603
6.299.2.1 DataResponseMarshaller . . . . .	1603
6.299.2.2 ~DataResponseMarshaller . . . . .	1603
6.299.3 Member Function Documentation . . . . .	1603
6.299.3.1 createObject . . . . .	1603
6.299.3.2 getDataStructureType . . . . .	1603
6.299.3.3 looseMarshal . . . . .	1603
6.299.3.4 looseUnmarshal . . . . .	1604
6.299.3.5 tightMarshal1 . . . . .	1604
6.299.3.6 tightMarshal2 . . . . .	1605
6.299.3.7 tightUnmarshal . . . . .	1605
6.300activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference . . . . .	1606
6.300.1 Detailed Description . . . . .	1606
6.300.2 Constructor & Destructor Documentation . . . . .	1607
6.300.2.1 DataResponseMarshaller . . . . .	1607
6.300.2.2 ~DataResponseMarshaller . . . . .	1607
6.300.3 Member Function Documentation . . . . .	1607
6.300.3.1 createObject . . . . .	1607
6.300.3.2 getDataStructureType . . . . .	1607
6.300.3.3 looseMarshal . . . . .	1607
6.300.3.4 looseUnmarshal . . . . .	1608
6.300.3.5 tightMarshal1 . . . . .	1608
6.300.3.6 tightMarshal2 . . . . .	1609
6.300.3.7 tightUnmarshal . . . . .	1609
6.301activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference . . . . .	1610
6.301.1 Detailed Description . . . . .	1610
6.301.2 Constructor & Destructor Documentation . . . . .	1611
6.301.2.1 ~DataStreamMarshaller . . . . .	1611
6.301.3 Member Function Documentation . . . . .	1611
6.301.3.1 createObject . . . . .	1611
6.301.3.2 getDataStructureType . . . . .	1617
6.301.3.3 looseMarshal . . . . .	1624
6.301.3.4 looseUnmarshal . . . . .	1631
6.301.3.5 tightMarshal1 . . . . .	1638
6.301.3.6 tightMarshal2 . . . . .	1645

6.301.3.7 tightUnmarshal . . . . .	1652
6.302activemq::commands::DataStructure Class Reference . . . . .	1660
6.302.1 Constructor & Destructor Documentation . . . . .	1660
6.302.1.1 ~DataStructure . . . . .	1660
6.302.2 Member Function Documentation . . . . .	1660
6.302.2.1 cloneDataStructure . . . . .	1660
6.302.2.2 copyDataStructure . . . . .	1661
6.302.2.3 equals . . . . .	1661
6.302.2.4 getDataStructureType . . . . .	1662
6.302.2.5 toString . . . . .	1663
6.303decaf::util::Date Class Reference . . . . .	1665
6.303.1 Detailed Description . . . . .	1666
6.303.2 Constructor & Destructor Documentation . . . . .	1666
6.303.2.1 Date . . . . .	1666
6.303.2.2 Date . . . . .	1666
6.303.2.3 Date . . . . .	1666
6.303.2.4 ~Date . . . . .	1666
6.303.3 Member Function Documentation . . . . .	1666
6.303.3.1 after . . . . .	1666
6.303.3.2 before . . . . .	1666
6.303.3.3 compareTo . . . . .	1667
6.303.3.4 equals . . . . .	1667
6.303.3.5 getTime . . . . .	1667
6.303.3.6 operator< . . . . .	1667
6.303.3.7 operator= . . . . .	1668
6.303.3.8 operator== . . . . .	1668
6.303.3.9 setTime . . . . .	1668
6.303.3.10 toString . . . . .	1668
6.304decaf::internal::DecafRuntime Class Reference . . . . .	1670
6.304.1 Detailed Description . . . . .	1670
6.304.2 Constructor & Destructor Documentation . . . . .	1670
6.304.2.1 DecafRuntime . . . . .	1670
6.304.2.2 ~DecafRuntime . . . . .	1670
6.304.3 Member Function Documentation . . . . .	1670
6.304.3.1 getGlobalPool . . . . .	1670
6.305activemq::threads::DedicatedTaskRunner Class Reference . . . . .	1671

6.305.1 Constructor & Destructor Documentation . . . . .	1671
6.305.1.1 DedicatedTaskRunner . . . . .	1671
6.305.1.2 ~DedicatedTaskRunner . . . . .	1671
6.305.2 Member Function Documentation . . . . .	1671
6.305.2.1 run . . . . .	1671
6.305.2.2 shutdown . . . . .	1672
6.305.2.3 shutdown . . . . .	1672
6.305.2.4 wakeup . . . . .	1672
6.306activemq::core::policies::DefaultPrefetchPolicy Class Reference . . . . .	1673
6.306.1 Constructor & Destructor Documentation . . . . .	1674
6.306.1.1 DefaultPrefetchPolicy . . . . .	1674
6.306.1.2 ~DefaultPrefetchPolicy . . . . .	1674
6.306.2 Member Function Documentation . . . . .	1674
6.306.2.1 clone . . . . .	1674
6.306.2.2 getDurableTopicPrefetch . . . . .	1674
6.306.2.3 getMaxPrefetchLimit . . . . .	1674
6.306.2.4 getQueueBrowserPrefetch . . . . .	1674
6.306.2.5 getQueuePrefetch . . . . .	1675
6.306.2.6 getTopicPrefetch . . . . .	1675
6.306.2.7 setDurableTopicPrefetch . . . . .	1675
6.306.2.8 setQueueBrowserPrefetch . . . . .	1675
6.306.2.9 setQueuePrefetch . . . . .	1676
6.306.2.10setTopicPrefetch . . . . .	1676
6.306.3 Field Documentation . . . . .	1676
6.306.3.1 DEFAULT_DURABLE_TOPIC_PREFETCH . . . . .	1676
6.306.3.2 DEFAULT_QUEUE_BROWSER_PREFETCH . . . . .	1676
6.306.3.3 DEFAULT_QUEUE_PREFETCH . . . . .	1676
6.306.3.4 DEFAULT_TOPIC_PREFETCH . . . . .	1676
6.306.3.5 MAX_PREFETCH_SIZE . . . . .	1676
6.307activemq::core::policies::DefaultRedeliveryPolicy Class Reference . . . . .	1677
6.307.1 Constructor & Destructor Documentation . . . . .	1678
6.307.1.1 DefaultRedeliveryPolicy . . . . .	1678
6.307.1.2 ~DefaultRedeliveryPolicy . . . . .	1678
6.307.2 Member Function Documentation . . . . .	1678
6.307.2.1 clone . . . . .	1678
6.307.2.2 getBackOffMultiplier . . . . .	1678

6.307.2.3	getCollisionAvoidancePercent . . . . .	1678
6.307.2.4	getInitialRedeliveryDelay . . . . .	1678
6.307.2.5	getMaximumRedeliveries . . . . .	1679
6.307.2.6	getRedeliveryDelay . . . . .	1679
6.307.2.7	isUseCollisionAvoidance . . . . .	1679
6.307.2.8	isUseExponentialBackOff . . . . .	1679
6.307.2.9	setBackOffMultiplier . . . . .	1679
6.307.2.10	setCollisionAvoidancePercent . . . . .	1680
6.307.2.11	setInitialRedeliveryDelay . . . . .	1680
6.307.2.12	setMaximumRedeliveries . . . . .	1680
6.307.2.13	setUseCollisionAvoidance . . . . .	1680
6.307.2.14	setUseExponentialBackOff . . . . .	1681
6.308	decaf::internal::net::DefaultServerSocketFactory Class Reference . . . . .	1682
6.308.1	Detailed Description . . . . .	1683
6.308.2	Constructor & Destructor Documentation . . . . .	1683
6.308.2.1	DefaultServerSocketFactory . . . . .	1683
6.308.2.2	~DefaultServerSocketFactory . . . . .	1683
6.308.3	Member Function Documentation . . . . .	1683
6.308.3.1	createServerSocket . . . . .	1683
6.308.3.2	createServerSocket . . . . .	1684
6.308.3.3	createServerSocket . . . . .	1684
6.308.3.4	createServerSocket . . . . .	1685
6.309	decaf::internal::net::DefaultSocketFactory Class Reference . . . . .	1686
6.309.1	Detailed Description . . . . .	1687
6.309.2	Constructor & Destructor Documentation . . . . .	1688
6.309.2.1	DefaultSocketFactory . . . . .	1688
6.309.2.2	~DefaultSocketFactory . . . . .	1688
6.309.3	Member Function Documentation . . . . .	1688
6.309.3.1	createSocket . . . . .	1688
6.309.3.2	createSocket . . . . .	1688
6.309.3.3	createSocket . . . . .	1689
6.309.3.4	createSocket . . . . .	1689
6.309.3.5	createSocket . . . . .	1690
6.310	decaf::internal::net::ssl::DefaultSSLContext Class Reference . . . . .	1691
6.310.1	Detailed Description . . . . .	1691
6.310.2	Constructor & Destructor Documentation . . . . .	1691

6.310.2.1 DefaultSSLContext . . . . .	1691
6.310.2.2 ~DefaultSSLContext . . . . .	1691
6.310.3 Member Function Documentation . . . . .	1691
6.310.3.1 getContext . . . . .	1691
6.311decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference . . . . .	1692
6.311.1 Detailed Description . . . . .	1693
6.311.2 Constructor & Destructor Documentation . . . . .	1694
6.311.2.1 DefaultSSLServerSocketFactory . . . . .	1694
6.311.2.2 ~DefaultSSLServerSocketFactory . . . . .	1694
6.311.3 Member Function Documentation . . . . .	1694
6.311.3.1 createServerSocket . . . . .	1694
6.311.3.2 createServerSocket . . . . .	1694
6.311.3.3 createServerSocket . . . . .	1695
6.311.3.4 createServerSocket . . . . .	1695
6.311.3.5 getDefaultCipherSuites . . . . .	1695
6.311.3.6 getSupportedCipherSuites . . . . .	1696
6.312decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference . . . . .	1697
6.312.1 Detailed Description . . . . .	1699
6.312.2 Constructor & Destructor Documentation . . . . .	1699
6.312.2.1 DefaultSSLSocketFactory . . . . .	1699
6.312.2.2 ~DefaultSSLSocketFactory . . . . .	1699
6.312.3 Member Function Documentation . . . . .	1699
6.312.3.1 createSocket . . . . .	1699
6.312.3.2 createSocket . . . . .	1700
6.312.3.3 createSocket . . . . .	1700
6.312.3.4 createSocket . . . . .	1701
6.312.3.5 createSocket . . . . .	1701
6.312.3.6 createSocket . . . . .	1702
6.312.3.7 getDefaultCipherSuites . . . . .	1702
6.312.3.8 getSupportedCipherSuites . . . . .	1703
6.313activemq::transport::DefaultTransportListener Class Reference . . . . .	1704
6.313.1 Constructor & Destructor Documentation . . . . .	1704
6.313.1.1 ~DefaultTransportListener . . . . .	1704
6.313.2 Member Function Documentation . . . . .	1704
6.313.2.1 onCommand . . . . .	1704
6.313.2.2 onException . . . . .	1704

6.313.2.3 transportInterrupted . . . . .	1705
6.313.2.4 transportResumed . . . . .	1705
6.314decaf::util::zip::Deflater Class Reference . . . . .	1706
6.314.1 Detailed Description . . . . .	1708
6.314.2 Constructor & Destructor Documentation . . . . .	1708
6.314.2.1 Deflater . . . . .	1708
6.314.2.2 Deflater . . . . .	1708
6.314.2.3 ~Deflater . . . . .	1709
6.314.3 Member Function Documentation . . . . .	1709
6.314.3.1 deflate . . . . .	1709
6.314.3.2 deflate . . . . .	1709
6.314.3.3 deflate . . . . .	1709
6.314.3.4 end . . . . .	1710
6.314.3.5 finish . . . . .	1710
6.314.3.6 finished . . . . .	1710
6.314.3.7 getAdler . . . . .	1710
6.314.3.8 getBytesRead . . . . .	1710
6.314.3.9 getBytesWritten . . . . .	1711
6.314.3.10needsInput . . . . .	1711
6.314.3.11reset . . . . .	1711
6.314.3.12setDictionary . . . . .	1711
6.314.3.13setDictionary . . . . .	1712
6.314.3.14setDictionary . . . . .	1712
6.314.3.15setInput . . . . .	1712
6.314.3.16setInput . . . . .	1713
6.314.3.17setInput . . . . .	1713
6.314.3.18setLevel . . . . .	1714
6.314.3.19setStrategy . . . . .	1714
6.314.4 Field Documentation . . . . .	1714
6.314.4.1 BEST_COMPRESSION . . . . .	1714
6.314.4.2 BEST_SPEED . . . . .	1714
6.314.4.3 DEFAULT_COMPRESSION . . . . .	1714
6.314.4.4 DEFAULT_STRATEGY . . . . .	1714
6.314.4.5 DEFLATED . . . . .	1715
6.314.4.6 FILTERED . . . . .	1715
6.314.4.7 HUFFMAN_ONLY . . . . .	1715



6.314.4.8 NO_COMPRESSION . . . . .	1715
6.315decaf::util::zip::DeflaterOutputStream Class Reference . . . . .	1716
6.315.1 Detailed Description . . . . .	1717
6.315.2 Constructor & Destructor Documentation . . . . .	1717
6.315.2.1 DeflaterOutputStream . . . . .	1717
6.315.2.2 DeflaterOutputStream . . . . .	1717
6.315.2.3 DeflaterOutputStream . . . . .	1718
6.315.2.4 ~DeflaterOutputStream . . . . .	1718
6.315.3 Member Function Documentation . . . . .	1718
6.315.3.1 close . . . . .	1718
6.315.3.2 deflate . . . . .	1718
6.315.3.3 doWriteArrayBounded . . . . .	1719
6.315.3.4 doWriteByte . . . . .	1719
6.315.3.5 finish . . . . .	1719
6.315.4 Field Documentation . . . . .	1719
6.315.4.1 buf . . . . .	1719
6.315.4.2 DEFAULT_BUFFER_SIZE . . . . .	1719
6.315.4.3 deflater . . . . .	1719
6.315.4.4 isDone . . . . .	1719
6.315.4.5 ownDeflater . . . . .	1719
6.316decaf::util::concurrent::Delayed Class Reference . . . . .	1720
6.316.1 Detailed Description . . . . .	1720
6.316.2 Constructor & Destructor Documentation . . . . .	1720
6.316.2.1 ~Delayed . . . . .	1720
6.316.3 Member Function Documentation . . . . .	1720
6.316.3.1 getDelay . . . . .	1720
6.317cms::DeliveryMode Class Reference . . . . .	1721
6.317.1 Detailed Description . . . . .	1721
6.317.2 Member Enumeration Documentation . . . . .	1721
6.317.2.1 DELIVERY_MODE . . . . .	1721
6.317.3 Constructor & Destructor Documentation . . . . .	1722
6.317.3.1 ~DeliveryMode . . . . .	1722
6.318cms::Destination Class Reference . . . . .	1723
6.318.1 Detailed Description . . . . .	1723
6.318.2 Member Enumeration Documentation . . . . .	1723
6.318.2.1 DestinationType . . . . .	1723

6.318.3 Constructor & Destructor Documentation . . . . .	1724
6.318.3.1 ~Destination . . . . .	1724
6.318.4 Member Function Documentation . . . . .	1724
6.318.4.1 clone . . . . .	1724
6.318.4.2 copy . . . . .	1724
6.318.4.3 getCMSProperties . . . . .	1724
6.318.4.4 getDestinationType . . . . .	1724
6.319activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . .	1726
6.319.1 Field Documentation . . . . .	1726
6.319.1.1 ANY_CHILD . . . . .	1726
6.319.1.2 ANY_DESCENDENT . . . . .	1726
6.320activemq::commands::DestinationInfo Class Reference . . . . .	1727
6.320.1 Constructor & Destructor Documentation . . . . .	1728
6.320.1.1 DestinationInfo . . . . .	1728
6.320.1.2 ~DestinationInfo . . . . .	1728
6.320.2 Member Function Documentation . . . . .	1728
6.320.2.1 cloneDataStructure . . . . .	1728
6.320.2.2 copyDataStructure . . . . .	1728
6.320.2.3 equals . . . . .	1728
6.320.2.4 getBrokerPath . . . . .	1729
6.320.2.5 getBrokerPath . . . . .	1729
6.320.2.6 getConnectionId . . . . .	1729
6.320.2.7 getConnectionId . . . . .	1729
6.320.2.8 getDataStructureType . . . . .	1729
6.320.2.9 getDestination . . . . .	1730
6.320.2.10getDestination . . . . .	1730
6.320.2.11getOperationType . . . . .	1730
6.320.2.12getTimeout . . . . .	1730
6.320.2.13setBrokerPath . . . . .	1730
6.320.2.14setConnectionId . . . . .	1730
6.320.2.15setDestination . . . . .	1730
6.320.2.16setOperationType . . . . .	1730
6.320.2.17setTimeout . . . . .	1730
6.320.2.18oString . . . . .	1730
6.320.2.19visit . . . . .	1730
6.320.3 Field Documentation . . . . .	1731

6.320.3.1 brokerPath . . . . .	1731
6.320.3.2 connectionId . . . . .	1731
6.320.3.3 destination . . . . .	1731
6.320.3.4 ID_DESTINATIONINFO . . . . .	1731
6.320.3.5 operationType . . . . .	1731
6.320.3.6 timeout . . . . .	1731
6.321activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference . . . . .	1732
6.321.1 Detailed Description . . . . .	1732
6.321.2 Constructor & Destructor Documentation . . . . .	1733
6.321.2.1 DestinationInfoMarshaller . . . . .	1733
6.321.2.2 ~DestinationInfoMarshaller . . . . .	1733
6.321.3 Member Function Documentation . . . . .	1733
6.321.3.1 createObject . . . . .	1733
6.321.3.2 getDataStructureType . . . . .	1733
6.321.3.3 looseMarshal . . . . .	1733
6.321.3.4 looseUnmarshal . . . . .	1734
6.321.3.5 tightMarshal1 . . . . .	1734
6.321.3.6 tightMarshal2 . . . . .	1734
6.321.3.7 tightUnmarshal . . . . .	1735
6.322activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference . . . . .	1736
6.322.1 Detailed Description . . . . .	1736
6.322.2 Constructor & Destructor Documentation . . . . .	1737
6.322.2.1 DestinationInfoMarshaller . . . . .	1737
6.322.2.2 ~DestinationInfoMarshaller . . . . .	1737
6.322.3 Member Function Documentation . . . . .	1737
6.322.3.1 createObject . . . . .	1737
6.322.3.2 getDataStructureType . . . . .	1737
6.322.3.3 looseMarshal . . . . .	1737
6.322.3.4 looseUnmarshal . . . . .	1738
6.322.3.5 tightMarshal1 . . . . .	1738
6.322.3.6 tightMarshal2 . . . . .	1738
6.322.3.7 tightUnmarshal . . . . .	1739
6.323activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference . . . . .	1740
6.323.1 Detailed Description . . . . .	1740

6.323.2 Constructor & Destructor Documentation . . . . .	1741
6.323.2.1 DestinationInfoMarshaller . . . . .	1741
6.323.2.2 ~DestinationInfoMarshaller . . . . .	1741
6.323.3 Member Function Documentation . . . . .	1741
6.323.3.1 createObject . . . . .	1741
6.323.3.2 getDataStructureType . . . . .	1741
6.323.3.3 looseMarshal . . . . .	1741
6.323.3.4 looseUnmarshal . . . . .	1742
6.323.3.5 tightMarshal1 . . . . .	1742
6.323.3.6 tightMarshal2 . . . . .	1742
6.323.3.7 tightUnmarshal . . . . .	1743
6.324activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference . . . . .	1744
6.324.1 Detailed Description . . . . .	1744
6.324.2 Constructor & Destructor Documentation . . . . .	1745
6.324.2.1 DestinationInfoMarshaller . . . . .	1745
6.324.2.2 ~DestinationInfoMarshaller . . . . .	1745
6.324.3 Member Function Documentation . . . . .	1745
6.324.3.1 createObject . . . . .	1745
6.324.3.2 getDataStructureType . . . . .	1745
6.324.3.3 looseMarshal . . . . .	1745
6.324.3.4 looseUnmarshal . . . . .	1746
6.324.3.5 tightMarshal1 . . . . .	1746
6.324.3.6 tightMarshal2 . . . . .	1746
6.324.3.7 tightUnmarshal . . . . .	1747
6.325activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference . . . . .	1748
6.325.1 Detailed Description . . . . .	1748
6.325.2 Constructor & Destructor Documentation . . . . .	1749
6.325.2.1 DestinationInfoMarshaller . . . . .	1749
6.325.2.2 ~DestinationInfoMarshaller . . . . .	1749
6.325.3 Member Function Documentation . . . . .	1749
6.325.3.1 createObject . . . . .	1749
6.325.3.2 getDataStructureType . . . . .	1749
6.325.3.3 looseMarshal . . . . .	1749
6.325.3.4 looseUnmarshal . . . . .	1750
6.325.3.5 tightMarshal1 . . . . .	1750

6.325.3.6 tightMarshal2 . . . . .	1750
6.325.3.7 tightUnmarshal . . . . .	1751
6.326activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference . . . . .	1752
6.326.1 Detailed Description . . . . .	1752
6.326.2 Constructor & Destructor Documentation . . . . .	1753
6.326.2.1 DestinationInfoMarshaller . . . . .	1753
6.326.2.2 ~DestinationInfoMarshaller . . . . .	1753
6.326.3 Member Function Documentation . . . . .	1753
6.326.3.1 createObject . . . . .	1753
6.326.3.2 getDataStructureType . . . . .	1753
6.326.3.3 looseMarshal . . . . .	1753
6.326.3.4 looseUnmarshal . . . . .	1754
6.326.3.5 tightMarshal1 . . . . .	1754
6.326.3.6 tightMarshal2 . . . . .	1754
6.326.3.7 tightUnmarshal . . . . .	1755
6.327activemq::cmsutil::DestinationResolver Class Reference . . . . .	1756
6.327.1 Detailed Description . . . . .	1756
6.327.2 Constructor & Destructor Documentation . . . . .	1756
6.327.2.1 ~DestinationResolver . . . . .	1756
6.327.3 Member Function Documentation . . . . .	1756
6.327.3.1 destroy . . . . .	1756
6.327.3.2 init . . . . .	1756
6.327.3.3 resolveDestinationName . . . . .	1757
6.328activemq::commands::DiscoveryEvent Class Reference . . . . .	1758
6.328.1 Constructor & Destructor Documentation . . . . .	1759
6.328.1.1 DiscoveryEvent . . . . .	1759
6.328.1.2 ~DiscoveryEvent . . . . .	1759
6.328.2 Member Function Documentation . . . . .	1759
6.328.2.1 cloneDataStructure . . . . .	1759
6.328.2.2 copyDataStructure . . . . .	1759
6.328.2.3 equals . . . . .	1759
6.328.2.4 getBrokerName . . . . .	1760
6.328.2.5 getBrokerName . . . . .	1760
6.328.2.6 getDataStructureType . . . . .	1760
6.328.2.7 getServiceName . . . . .	1760
6.328.2.8 getServiceName . . . . .	1760

6.328.2.9	setBrokerName . . . . .	1760
6.328.2.10	getServiceName . . . . .	1760
6.328.2.11	toString . . . . .	1760
6.328.3	Field Documentation . . . . .	1761
6.328.3.1	brokerName . . . . .	1761
6.328.3.2	ID_DISCOVERYEVENT . . . . .	1761
6.328.3.3	serviceName . . . . .	1761
6.329	activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller Class Reference . . . . .	1762
6.329.1	Detailed Description . . . . .	1762
6.329.2	Constructor & Destructor Documentation . . . . .	1763
6.329.2.1	DiscoveryEventMarshaller . . . . .	1763
6.329.2.2	~DiscoveryEventMarshaller . . . . .	1763
6.329.3	Member Function Documentation . . . . .	1763
6.329.3.1	createObject . . . . .	1763
6.329.3.2	getDataStructureType . . . . .	1763
6.329.3.3	looseMarshal . . . . .	1763
6.329.3.4	looseUnmarshal . . . . .	1764
6.329.3.5	tightMarshal1 . . . . .	1764
6.329.3.6	tightMarshal2 . . . . .	1764
6.329.3.7	tightUnmarshal . . . . .	1765
6.330	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference . . . . .	1766
6.330.1	Detailed Description . . . . .	1766
6.330.2	Constructor & Destructor Documentation . . . . .	1767
6.330.2.1	DiscoveryEventMarshaller . . . . .	1767
6.330.2.2	~DiscoveryEventMarshaller . . . . .	1767
6.330.3	Member Function Documentation . . . . .	1767
6.330.3.1	createObject . . . . .	1767
6.330.3.2	getDataStructureType . . . . .	1767
6.330.3.3	looseMarshal . . . . .	1767
6.330.3.4	looseUnmarshal . . . . .	1768
6.330.3.5	tightMarshal1 . . . . .	1768
6.330.3.6	tightMarshal2 . . . . .	1768
6.330.3.7	tightUnmarshal . . . . .	1769
6.331	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference . . . . .	1770

6.331.1 Detailed Description . . . . .	1770
6.331.2 Constructor & Destructor Documentation . . . . .	1771
6.331.2.1 DiscoveryEventMarshaller . . . . .	1771
6.331.2.2 ~DiscoveryEventMarshaller . . . . .	1771
6.331.3 Member Function Documentation . . . . .	1771
6.331.3.1 createObject . . . . .	1771
6.331.3.2 getDataStructureType . . . . .	1771
6.331.3.3 looseMarshal . . . . .	1771
6.331.3.4 looseUnmarshal . . . . .	1772
6.331.3.5 tightMarshal1 . . . . .	1772
6.331.3.6 tightMarshal2 . . . . .	1772
6.331.3.7 tightUnmarshal . . . . .	1773
6.332activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference . . . . .	1774
6.332.1 Detailed Description . . . . .	1774
6.332.2 Constructor & Destructor Documentation . . . . .	1775
6.332.2.1 DiscoveryEventMarshaller . . . . .	1775
6.332.2.2 ~DiscoveryEventMarshaller . . . . .	1775
6.332.3 Member Function Documentation . . . . .	1775
6.332.3.1 createObject . . . . .	1775
6.332.3.2 getDataStructureType . . . . .	1775
6.332.3.3 looseMarshal . . . . .	1775
6.332.3.4 looseUnmarshal . . . . .	1776
6.332.3.5 tightMarshal1 . . . . .	1776
6.332.3.6 tightMarshal2 . . . . .	1776
6.332.3.7 tightUnmarshal . . . . .	1777
6.333activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference . . . . .	1778
6.333.1 Detailed Description . . . . .	1778
6.333.2 Constructor & Destructor Documentation . . . . .	1779
6.333.2.1 DiscoveryEventMarshaller . . . . .	1779
6.333.2.2 ~DiscoveryEventMarshaller . . . . .	1779
6.333.3 Member Function Documentation . . . . .	1779
6.333.3.1 createObject . . . . .	1779
6.333.3.2 getDataStructureType . . . . .	1779
6.333.3.3 looseMarshal . . . . .	1779
6.333.3.4 looseUnmarshal . . . . .	1780

6.333.3.5 tightMarshal1 . . . . .	1780
6.333.3.6 tightMarshal2 . . . . .	1780
6.333.3.7 tightUnmarshal . . . . .	1781
6.334activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference . . . . .	1782
6.334.1 Detailed Description . . . . .	1782
6.334.2 Constructor & Destructor Documentation . . . . .	1783
6.334.2.1 DiscoveryEventMarshaller . . . . .	1783
6.334.2.2 ~DiscoveryEventMarshaller . . . . .	1783
6.334.3 Member Function Documentation . . . . .	1783
6.334.3.1 createObject . . . . .	1783
6.334.3.2 getDataStructureType . . . . .	1783
6.334.3.3 looseMarshal . . . . .	1783
6.334.3.4 looseUnmarshal . . . . .	1784
6.334.3.5 tightMarshal1 . . . . .	1784
6.334.3.6 tightMarshal2 . . . . .	1784
6.334.3.7 tightUnmarshal . . . . .	1785
6.335activemq::core::DispatchData Class Reference . . . . .	1786
6.335.1 Detailed Description . . . . .	1786
6.335.2 Constructor & Destructor Documentation . . . . .	1786
6.335.2.1 DispatchData . . . . .	1786
6.335.2.2 DispatchData . . . . .	1786
6.335.3 Member Function Documentation . . . . .	1786
6.335.3.1 getConsumerId . . . . .	1786
6.335.3.2 getMessage . . . . .	1786
6.336activemq::core::Dispatcher Class Reference . . . . .	1787
6.336.1 Detailed Description . . . . .	1787
6.336.2 Constructor & Destructor Documentation . . . . .	1787
6.336.2.1 ~Dispatcher . . . . .	1787
6.336.3 Member Function Documentation . . . . .	1787
6.336.3.1 dispatch . . . . .	1787
6.337decaf::lang::Double Class Reference . . . . .	1788
6.337.1 Constructor & Destructor Documentation . . . . .	1790
6.337.1.1 Double . . . . .	1790
6.337.1.2 Double . . . . .	1790
6.337.1.3 ~Double . . . . .	1790
6.337.2 Member Function Documentation . . . . .	1790



6.337.2.1 byteValue . . . . .	1790
6.337.2.2 compare . . . . .	1790
6.337.2.3 compareTo . . . . .	1791
6.337.2.4 compareTo . . . . .	1791
6.337.2.5 doubleToLongBits . . . . .	1791
6.337.2.6 doubleToRawLongBits . . . . .	1792
6.337.2.7 doubleValue . . . . .	1792
6.337.2.8 equals . . . . .	1793
6.337.2.9 equals . . . . .	1793
6.337.2.10 float Value . . . . .	1793
6.337.2.11 int Value . . . . .	1793
6.337.2.12 isInfinite . . . . .	1793
6.337.2.13 isInfinite . . . . .	1794
6.337.2.14 isNaN . . . . .	1794
6.337.2.15 isNaN . . . . .	1794
6.337.2.16 longBitsToDouble . . . . .	1794
6.337.2.17 long Value . . . . .	1794
6.337.2.18 operator< . . . . .	1795
6.337.2.19 operator< . . . . .	1795
6.337.2.20 operator== . . . . .	1795
6.337.2.21 operator== . . . . .	1795
6.337.2.22 parseDouble . . . . .	1796
6.337.2.23 short Value . . . . .	1796
6.337.2.24 toHexString . . . . .	1796
6.337.2.25 toString . . . . .	1797
6.337.2.26 toString . . . . .	1797
6.337.2.27 valueOf . . . . .	1797
6.337.2.28 valueOf . . . . .	1798
6.337.3 Field Documentation . . . . .	1798
6.337.3.1 MAX_VALUE . . . . .	1798
6.337.3.2 MIN_VALUE . . . . .	1798
6.337.3.3 NaN . . . . .	1798
6.337.3.4 NEGATIVE_INFINITY . . . . .	1798
6.337.3.5 POSITIVE_INFINITY . . . . .	1798
6.337.3.6 SIZE . . . . .	1798
6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference . . . . .	1799

6.338.1 Constructor & Destructor Documentation . . . . .	1802
6.338.1.1 DoubleArrayBuffer . . . . .	1802
6.338.1.2 DoubleArrayBuffer . . . . .	1802
6.338.1.3 DoubleArrayBuffer . . . . .	1803
6.338.1.4 DoubleArrayBuffer . . . . .	1803
6.338.1.5 ~DoubleArrayBuffer . . . . .	1803
6.338.2 Member Function Documentation . . . . .	1803
6.338.2.1 array . . . . .	1803
6.338.2.2 arrayOffset . . . . .	1804
6.338.2.3 asReadOnlyBuffer . . . . .	1804
6.338.2.4 compact . . . . .	1805
6.338.2.5 duplicate . . . . .	1805
6.338.2.6 get . . . . .	1805
6.338.2.7 get . . . . .	1806
6.338.2.8 hasArray . . . . .	1806
6.338.2.9 isReadOnly . . . . .	1806
6.338.2.10put . . . . .	1806
6.338.2.11put . . . . .	1807
6.338.2.12setReadOnly . . . . .	1807
6.338.2.13lice . . . . .	1807
6.339decaf::nio::DoubleBuffer Class Reference . . . . .	1809
6.339.1 Detailed Description . . . . .	1811
6.339.2 Constructor & Destructor Documentation . . . . .	1811
6.339.2.1 DoubleBuffer . . . . .	1811
6.339.2.2 ~DoubleBuffer . . . . .	1811
6.339.3 Member Function Documentation . . . . .	1811
6.339.3.1 allocate . . . . .	1811
6.339.3.2 array . . . . .	1812
6.339.3.3 arrayOffset . . . . .	1812
6.339.3.4 asReadOnlyBuffer . . . . .	1813
6.339.3.5 compact . . . . .	1813
6.339.3.6 compareTo . . . . .	1813
6.339.3.7 duplicate . . . . .	1813
6.339.3.8 equals . . . . .	1814
6.339.3.9 get . . . . .	1814
6.339.3.10get . . . . .	1814

6.339.3.11	get . . . . .	1815
6.339.3.12	get . . . . .	1815
6.339.3.13	hasArray . . . . .	1815
6.339.3.14	operator< . . . . .	1816
6.339.3.15	operator== . . . . .	1816
6.339.3.16	put . . . . .	1816
6.339.3.17	put . . . . .	1816
6.339.3.18	put . . . . .	1816
6.339.3.19	put . . . . .	1817
6.339.3.20	put . . . . .	1817
6.339.3.21	slice . . . . .	1818
6.339.3.22	toString . . . . .	1818
6.339.3.23	wrap . . . . .	1818
6.339.3.24	wrap . . . . .	1819
6.340	decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference . . . . .	1820
6.341	activemq::cmsutil::DynamicDestinationResolver Class Reference . . . . .	1821
6.341.1	Detailed Description . . . . .	1821
6.341.2	Constructor & Destructor Documentation . . . . .	1822
6.341.2.1	DynamicDestinationResolver . . . . .	1822
6.341.2.2	DynamicDestinationResolver . . . . .	1822
6.341.2.3	~DynamicDestinationResolver . . . . .	1822
6.341.3	Member Function Documentation . . . . .	1822
6.341.3.1	destroy . . . . .	1822
6.341.3.2	init . . . . .	1822
6.341.3.3	operator= . . . . .	1822
6.341.3.4	resolveDestinationName . . . . .	1822
6.342	decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference . . . . .	1824
6.342.1	Constructor & Destructor Documentation . . . . .	1824
6.342.1.1	Entry . . . . .	1824
6.342.1.2	~Entry . . . . .	1824
6.342.2	Member Function Documentation . . . . .	1824
6.342.2.1	getKey . . . . .	1824
6.342.2.2	getValue . . . . .	1824
6.342.2.3	setValue . . . . .	1824
6.343	decaf::io::EOFException Class Reference . . . . .	1825
6.343.1	Constructor & Destructor Documentation . . . . .	1825

6.343.1.1 EOFException . . . . .	1825
6.343.1.2 EOFException . . . . .	1825
6.343.1.3 EOFException . . . . .	1826
6.343.1.4 EOFException . . . . .	1826
6.343.1.5 EOFException . . . . .	1826
6.343.1.6 EOFException . . . . .	1826
6.343.1.7 ~EOFException . . . . .	1827
6.343.2 Member Function Documentation . . . . .	1827
6.343.2.1 clone . . . . .	1827
6.344decaf::util::logging::ErrorManager Class Reference . . . . .	1828
6.344.1 Detailed Description . . . . .	1828
6.344.2 Constructor & Destructor Documentation . . . . .	1829
6.344.2.1 ErrorManager . . . . .	1829
6.344.2.2 ~ErrorManager . . . . .	1829
6.344.3 Member Function Documentation . . . . .	1829
6.344.3.1 error . . . . .	1829
6.344.4 Field Documentation . . . . .	1829
6.344.4.1 CLOSE_FAILURE . . . . .	1829
6.344.4.2 FLUSH_FAILURE . . . . .	1829
6.344.4.3 FORMAT_FAILURE . . . . .	1829
6.344.4.4 GENERIC_FAILURE . . . . .	1829
6.344.4.5 OPEN_FAILURE . . . . .	1829
6.344.4.6 WRITE_FAILURE . . . . .	1830
6.345decaf::lang::Exception Class Reference . . . . .	1831
6.345.1 Constructor & Destructor Documentation . . . . .	1832
6.345.1.1 Exception . . . . .	1832
6.345.1.2 Exception . . . . .	1833
6.345.1.3 Exception . . . . .	1833
6.345.1.4 Exception . . . . .	1833
6.345.1.5 Exception . . . . .	1833
6.345.1.6 ~Exception . . . . .	1834
6.345.2 Member Function Documentation . . . . .	1834
6.345.2.1 buildMessage . . . . .	1834
6.345.2.2 clone . . . . .	1834
6.345.2.3 getCause . . . . .	1835
6.345.2.4 getMessage . . . . .	1835

6.345.2.5	getStackTrace . . . . .	1835
6.345.2.6	getStackTraceString . . . . .	1835
6.345.2.7	initCause . . . . .	1836
6.345.2.8	operator= . . . . .	1836
6.345.2.9	printStackTrace . . . . .	1836
6.345.2.10	printStackTrace . . . . .	1836
6.345.2.11	setMark . . . . .	1836
6.345.2.12	setMessage . . . . .	1837
6.345.2.13	setStackTrace . . . . .	1837
6.345.2.14	what . . . . .	1837
6.345.3	Field Documentation . . . . .	1837
6.345.3.1	cause . . . . .	1837
6.345.3.2	message . . . . .	1837
6.345.3.3	stackTrace . . . . .	1837
6.346	cms::ExceptionListener Class Reference . . . . .	1838
6.346.1	Detailed Description . . . . .	1838
6.346.2	Constructor & Destructor Documentation . . . . .	1838
6.346.2.1	~ExceptionListener . . . . .	1838
6.346.3	Member Function Documentation . . . . .	1838
6.346.3.1	onException . . . . .	1838
6.347	activemq::commands::ExceptionResponse Class Reference . . . . .	1839
6.347.1	Constructor & Destructor Documentation . . . . .	1840
6.347.1.1	ExceptionResponse . . . . .	1840
6.347.1.2	~ExceptionResponse . . . . .	1840
6.347.2	Member Function Documentation . . . . .	1840
6.347.2.1	cloneDataStructure . . . . .	1840
6.347.2.2	copyDataStructure . . . . .	1840
6.347.2.3	equals . . . . .	1840
6.347.2.4	getDataStructureType . . . . .	1840
6.347.2.5	getException . . . . .	1841
6.347.2.6	getException . . . . .	1841
6.347.2.7	setException . . . . .	1841
6.347.2.8	toString . . . . .	1841
6.347.3	Field Documentation . . . . .	1841
6.347.3.1	exception . . . . .	1841
6.347.3.2	ID_EXCEPTIONRESPONSE . . . . .	1841

6.348	activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	Class	
	Reference		1842
6.348.1	Detailed Description		1842
6.348.2	Constructor & Destructor Documentation		1843
	6.348.2.1 ExceptionResponseMarshaller		1843
	6.348.2.2 ~ExceptionResponseMarshaller		1843
6.348.3	Member Function Documentation		1843
	6.348.3.1 createObject		1843
	6.348.3.2 getDataStructureType		1843
	6.348.3.3 looseMarshal		1843
	6.348.3.4 looseUnmarshal		1844
	6.348.3.5 tightMarshal1		1844
	6.348.3.6 tightMarshal2		1845
	6.348.3.7 tightUnmarshal		1845
6.349	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	Class	
	Reference		1846
6.349.1	Detailed Description		1846
6.349.2	Constructor & Destructor Documentation		1847
	6.349.2.1 ExceptionResponseMarshaller		1847
	6.349.2.2 ~ExceptionResponseMarshaller		1847
6.349.3	Member Function Documentation		1847
	6.349.3.1 createObject		1847
	6.349.3.2 getDataStructureType		1847
	6.349.3.3 looseMarshal		1847
	6.349.3.4 looseUnmarshal		1848
	6.349.3.5 tightMarshal1		1848
	6.349.3.6 tightMarshal2		1849
	6.349.3.7 tightUnmarshal		1849
6.350	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	Class	
	Reference		1850
6.350.1	Detailed Description		1850
6.350.2	Constructor & Destructor Documentation		1851
	6.350.2.1 ExceptionResponseMarshaller		1851
	6.350.2.2 ~ExceptionResponseMarshaller		1851
6.350.3	Member Function Documentation		1851
	6.350.3.1 createObject		1851
	6.350.3.2 getDataStructureType		1851

6.350.3.3 looseMarshal . . . . .	1851
6.350.3.4 looseUnmarshal . . . . .	1852
6.350.3.5 tightMarshal1 . . . . .	1852
6.350.3.6 tightMarshal2 . . . . .	1853
6.350.3.7 tightUnmarshal . . . . .	1853
6.351activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class	
Reference . . . . .	1854
6.351.1 Detailed Description . . . . .	1854
6.351.2 Constructor & Destructor Documentation . . . . .	1855
6.351.2.1 ExceptionResponseMarshaller . . . . .	1855
6.351.2.2 ~ExceptionResponseMarshaller . . . . .	1855
6.351.3 Member Function Documentation . . . . .	1855
6.351.3.1 createObject . . . . .	1855
6.351.3.2 getDataStructureType . . . . .	1855
6.351.3.3 looseMarshal . . . . .	1855
6.351.3.4 looseUnmarshal . . . . .	1856
6.351.3.5 tightMarshal1 . . . . .	1856
6.351.3.6 tightMarshal2 . . . . .	1857
6.351.3.7 tightUnmarshal . . . . .	1857
6.352activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class	
Reference . . . . .	1858
6.352.1 Detailed Description . . . . .	1858
6.352.2 Constructor & Destructor Documentation . . . . .	1859
6.352.2.1 ExceptionResponseMarshaller . . . . .	1859
6.352.2.2 ~ExceptionResponseMarshaller . . . . .	1859
6.352.3 Member Function Documentation . . . . .	1859
6.352.3.1 createObject . . . . .	1859
6.352.3.2 getDataStructureType . . . . .	1859
6.352.3.3 looseMarshal . . . . .	1859
6.352.3.4 looseUnmarshal . . . . .	1860
6.352.3.5 tightMarshal1 . . . . .	1860
6.352.3.6 tightMarshal2 . . . . .	1861
6.352.3.7 tightUnmarshal . . . . .	1861
6.353activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class	
Reference . . . . .	1862
6.353.1 Detailed Description . . . . .	1862
6.353.2 Constructor & Destructor Documentation . . . . .	1863

6.353.2.1	ExceptionResponseMarshaller	1863
6.353.2.2	~ExceptionResponseMarshaller	1863
6.353.3	Member Function Documentation	1863
6.353.3.1	createObject	1863
6.353.3.2	getDataStructureType	1863
6.353.3.3	looseMarshal	1863
6.353.3.4	looseUnmarshal	1864
6.353.3.5	tightMarshal1	1864
6.353.3.6	tightMarshal2	1865
6.353.3.7	tightUnmarshal	1865
6.354	decaf::util::concurrent::ExecutionException Class Reference	1866
6.354.1	Constructor & Destructor Documentation	1866
6.354.1.1	ExecutionException	1866
6.354.1.2	ExecutionException	1866
6.354.1.3	ExecutionException	1867
6.354.1.4	ExecutionException	1867
6.354.1.5	ExecutionException	1867
6.354.1.6	ExecutionException	1867
6.354.1.7	~ExecutionException	1868
6.354.2	Member Function Documentation	1868
6.354.2.1	clone	1868
6.355	decaf::util::concurrent::Executor Class Reference	1869
6.355.1	Detailed Description	1869
6.355.2	Constructor & Destructor Documentation	1870
6.355.2.1	~Executor	1870
6.355.3	Member Function Documentation	1870
6.355.3.1	execute	1870
6.356	decaf::util::concurrent::ExecutorService Class Reference	1871
6.356.1	Detailed Description	1871
6.356.2	Constructor & Destructor Documentation	1872
6.356.2.1	~ExecutorService	1872
6.356.3	Member Function Documentation	1872
6.356.3.1	awaitTermination	1872
6.357	activemq::transport::failover::FailoverTransport Class Reference	1873
6.357.1	Constructor & Destructor Documentation	1875
6.357.1.1	FailoverTransport	1875



6.357.1.2 ~FailoverTransport . . . . .	1875
6.357.2 Member Function Documentation . . . . .	1875
6.357.2.1 add . . . . .	1875
6.357.2.2 addURI . . . . .	1876
6.357.2.3 close . . . . .	1876
6.357.2.4 getBackOffMultiplier . . . . .	1877
6.357.2.5 getBackupPoolSize . . . . .	1877
6.357.2.6 getInitialReconnectDelay . . . . .	1877
6.357.2.7 getMaxCacheSize . . . . .	1877
6.357.2.8 getMaxReconnectAttempts . . . . .	1877
6.357.2.9 getMaxReconnectDelay . . . . .	1877
6.357.2.10getReconnectDelay . . . . .	1877
6.357.2.11getRemoteAddress . . . . .	1877
6.357.2.12getStartupMaxReconnectAttempts . . . . .	1877
6.357.2.13getTimeout . . . . .	1877
6.357.2.14getTransportListener . . . . .	1877
6.357.2.15handleTransportFailure . . . . .	1878
6.357.2.16sBackup . . . . .	1878
6.357.2.17sClosed . . . . .	1878
6.357.2.18sConnected . . . . .	1878
6.357.2.19sFaultTolerant . . . . .	1878
6.357.2.20sInitialized . . . . .	1879
6.357.2.21sPending . . . . .	1879
6.357.2.22sRandomize . . . . .	1879
6.357.2.23sTrackMessages . . . . .	1879
6.357.2.24sTrackTransactionProducers . . . . .	1879
6.357.2.25sUseExponentialBackOff . . . . .	1879
6.357.2.26terate . . . . .	1879
6.357.2.27narrow . . . . .	1879
6.357.2.28neway . . . . .	1880
6.357.2.29reconnect . . . . .	1880
6.357.2.30reconnect . . . . .	1880
6.357.2.31removeURI . . . . .	1881
6.357.2.32request . . . . .	1881
6.357.2.33request . . . . .	1881
6.357.2.34estoreTransport . . . . .	1882

6.357.2.35	setBackOffMultiplier . . . . .	1882
6.357.2.36	setBackup . . . . .	1882
6.357.2.37	setBackupPoolSize . . . . .	1882
6.357.2.38	setConnectionInterruptProcessingComplete . . . . .	1882
6.357.2.39	setInitialized . . . . .	1882
6.357.2.40	setInitialReconnectDelay . . . . .	1883
6.357.2.41	setMaxCacheSize . . . . .	1883
6.357.2.42	setMaxReconnectAttempts . . . . .	1883
6.357.2.43	setMaxReconnectDelay . . . . .	1883
6.357.2.44	setRandomize . . . . .	1883
6.357.2.45	setReconnectDelay . . . . .	1883
6.357.2.46	setStartupMaxReconnectAttempts . . . . .	1883
6.357.2.47	setTimeout . . . . .	1883
6.357.2.48	setTrackMessages . . . . .	1883
6.357.2.49	setTrackTransactionProducers . . . . .	1883
6.357.2.50	setTransportListener . . . . .	1883
6.357.2.51	setUseExponentialBackOff . . . . .	1884
6.357.2.52	setWireFormat . . . . .	1884
6.357.2.53	start . . . . .	1884
6.357.2.54	stop . . . . .	1884
6.357.3	Friends And Related Function Documentation . . . . .	1884
6.357.3.1	FailoverTransportListener . . . . .	1884
6.358	activemq::transport::failover::FailoverTransportFactory Class Reference . . . . .	1885
6.358.1	Detailed Description . . . . .	1885
6.358.2	Constructor & Destructor Documentation . . . . .	1886
6.358.2.1	~FailoverTransportFactory . . . . .	1886
6.358.3	Member Function Documentation . . . . .	1886
6.358.3.1	create . . . . .	1886
6.358.3.2	createComposite . . . . .	1886
6.358.3.3	doCreateComposite . . . . .	1886
6.359	activemq::transport::failover::FailoverTransportListener Class Reference . . . . .	1888
6.359.1	Detailed Description . . . . .	1888
6.359.2	Constructor & Destructor Documentation . . . . .	1889
6.359.2.1	FailoverTransportListener . . . . .	1889
6.359.2.2	~FailoverTransportListener . . . . .	1889
6.359.3	Member Function Documentation . . . . .	1889

6.359.3.1 onCommand . . . . .	1889
6.359.3.2 onException . . . . .	1889
6.359.3.3 transportInterrupted . . . . .	1889
6.359.3.4 transportResumed . . . . .	1889
6.360decaf::io::FileDescriptor Class Reference . . . . .	1891
6.360.1 Detailed Description . . . . .	1891
6.360.2 Constructor & Destructor Documentation . . . . .	1892
6.360.2.1 FileDescriptor . . . . .	1892
6.360.2.2 FileDescriptor . . . . .	1892
6.360.2.3 ~FileDescriptor . . . . .	1892
6.360.3 Member Function Documentation . . . . .	1892
6.360.3.1 sync . . . . .	1892
6.360.3.2 valid . . . . .	1892
6.360.4 Field Documentation . . . . .	1892
6.360.4.1 descriptor . . . . .	1892
6.360.4.2 err . . . . .	1892
6.360.4.3 in . . . . .	1892
6.360.4.4 out . . . . .	1892
6.360.4.5 readonly . . . . .	1892
6.361decaf::util::logging::Filter Class Reference . . . . .	1893
6.361.1 Detailed Description . . . . .	1893
6.361.2 Constructor & Destructor Documentation . . . . .	1893
6.361.2.1 ~Filter . . . . .	1893
6.361.3 Member Function Documentation . . . . .	1893
6.361.3.1 isLoggable . . . . .	1893
6.362decaf::io::FilterInputStream Class Reference . . . . .	1894
6.362.1 Detailed Description . . . . .	1896
6.362.2 Constructor & Destructor Documentation . . . . .	1896
6.362.2.1 FilterInputStream . . . . .	1896
6.362.2.2 ~FilterInputStream . . . . .	1896
6.362.3 Member Function Documentation . . . . .	1896
6.362.3.1 available . . . . .	1896
6.362.3.2 close . . . . .	1897
6.362.3.3 doReadArray . . . . .	1897
6.362.3.4 doReadArrayBounded . . . . .	1897
6.362.3.5 doReadByte . . . . .	1897

6.362.3.6	isClosed . . . . .	1897
6.362.3.7	mark . . . . .	1897
6.362.3.8	markSupported . . . . .	1898
6.362.3.9	reset . . . . .	1898
6.362.3.10	skip . . . . .	1899
6.362.4	Field Documentation . . . . .	1899
6.362.4.1	closed . . . . .	1899
6.362.4.2	inputStream . . . . .	1899
6.362.4.3	own . . . . .	1899
6.363	decaf::io::FilterOutputStream Class Reference . . . . .	1900
6.363.1	Detailed Description . . . . .	1901
6.363.2	Constructor & Destructor Documentation . . . . .	1901
6.363.2.1	FilterOutputStream . . . . .	1901
6.363.2.2	~FilterOutputStream . . . . .	1901
6.363.3	Member Function Documentation . . . . .	1901
6.363.3.1	close . . . . .	1901
6.363.3.2	doWriteArray . . . . .	1902
6.363.3.3	doWriteArrayBounded . . . . .	1902
6.363.3.4	doWriteByte . . . . .	1902
6.363.3.5	flush . . . . .	1902
6.363.3.6	isClosed . . . . .	1902
6.363.3.7	toString . . . . .	1903
6.363.4	Field Documentation . . . . .	1903
6.363.4.1	closed . . . . .	1903
6.363.4.2	outputStream . . . . .	1903
6.363.4.3	own . . . . .	1903
6.364	decaf::lang::Float Class Reference . . . . .	1904
6.364.1	Constructor & Destructor Documentation . . . . .	1906
6.364.1.1	Float . . . . .	1906
6.364.1.2	Float . . . . .	1906
6.364.1.3	Float . . . . .	1906
6.364.1.4	~Float . . . . .	1906
6.364.2	Member Function Documentation . . . . .	1906
6.364.2.1	byteValue . . . . .	1906
6.364.2.2	compare . . . . .	1907
6.364.2.3	compareTo . . . . .	1907

6.364.2.4	compareTo	1907
6.364.2.5	doubleValue	1907
6.364.2.6	equals	1908
6.364.2.7	equals	1908
6.364.2.8	floatToIntBits	1908
6.364.2.9	floatToRawIntBits	1908
6.364.2.10	floatValue	1909
6.364.2.11	intBitsToFloat	1909
6.364.2.12	intValue	1909
6.364.2.13	isInfinite	1910
6.364.2.14	isInfinite	1910
6.364.2.15	isNaN	1910
6.364.2.16	isNaN	1910
6.364.2.17	longValue	1910
6.364.2.18	operator<	1910
6.364.2.19	operator<	1911
6.364.2.20	operator==	1911
6.364.2.21	operator==	1911
6.364.2.22	parseFloat	1911
6.364.2.23	shortValue	1912
6.364.2.24	toHexString	1912
6.364.2.25	toString	1913
6.364.2.26	toString	1913
6.364.2.27	valueOf	1913
6.364.2.28	valueOf	1914
6.364.3	Field Documentation	1914
6.364.3.1	MAX_VALUE	1914
6.364.3.2	MIN_VALUE	1914
6.364.3.3	NaN	1914
6.364.3.4	NEGATIVE_INFINITY	1914
6.364.3.5	POSITIVE_INFINITY	1914
6.364.3.6	SIZE	1914
6.365	decaf::internal::nio::FloatArrayBuffer Class Reference	1915
6.365.1	Constructor & Destructor Documentation	1918
6.365.1.1	FloatArrayBuffer	1918
6.365.1.2	FloatArrayBuffer	1918

6.365.1.3 FloatArrayBuffer . . . . .	1919
6.365.1.4 FloatArrayBuffer . . . . .	1919
6.365.1.5 ~FloatArrayBuffer . . . . .	1919
6.365.2 Member Function Documentation . . . . .	1919
6.365.2.1 array . . . . .	1919
6.365.2.2 arrayOffset . . . . .	1920
6.365.2.3 asReadOnlyBuffer . . . . .	1920
6.365.2.4 compact . . . . .	1921
6.365.2.5 duplicate . . . . .	1921
6.365.2.6 get . . . . .	1921
6.365.2.7 get . . . . .	1922
6.365.2.8 hasArray . . . . .	1922
6.365.2.9 isReadOnly . . . . .	1922
6.365.2.10put . . . . .	1922
6.365.2.11put . . . . .	1923
6.365.2.12setReadOnly . . . . .	1923
6.365.2.13slice . . . . .	1923
6.366decaf::nio::FloatBuffer Class Reference . . . . .	1925
6.366.1 Detailed Description . . . . .	1927
6.366.2 Constructor & Destructor Documentation . . . . .	1927
6.366.2.1 FloatBuffer . . . . .	1927
6.366.2.2 ~FloatBuffer . . . . .	1927
6.366.3 Member Function Documentation . . . . .	1927
6.366.3.1 allocate . . . . .	1927
6.366.3.2 array . . . . .	1928
6.366.3.3 arrayOffset . . . . .	1928
6.366.3.4 asReadOnlyBuffer . . . . .	1928
6.366.3.5 compact . . . . .	1929
6.366.3.6 compareTo . . . . .	1929
6.366.3.7 duplicate . . . . .	1929
6.366.3.8 equals . . . . .	1929
6.366.3.9 get . . . . .	1929
6.366.3.10get . . . . .	1930
6.366.3.11get . . . . .	1930
6.366.3.12get . . . . .	1931
6.366.3.13hasArray . . . . .	1931

6.366.3.14	operator<	1931
6.366.3.15	operator==	1931
6.366.3.16	put	1931
6.366.3.17	put	1932
6.366.3.18	put	1932
6.366.3.19	put	1932
6.366.3.20	put	1933
6.366.3.21	slice	1934
6.366.3.22	toString	1934
6.366.3.23	wrap	1934
6.366.3.24	wrap	1934
6.367	decaf::io::Flushable Class Reference	1936
6.367.1	Detailed Description	1936
6.367.2	Constructor & Destructor Documentation	1936
6.367.2.1	~Flushable	1936
6.367.3	Member Function Documentation	1936
6.367.3.1	flush	1936
6.368	activemq::commands::FlushCommand Class Reference	1937
6.368.1	Constructor & Destructor Documentation	1938
6.368.1.1	FlushCommand	1938
6.368.1.2	~FlushCommand	1938
6.368.2	Member Function Documentation	1938
6.368.2.1	cloneDataStructure	1938
6.368.2.2	copyDataStructure	1938
6.368.2.3	equals	1938
6.368.2.4	getDataStructureType	1938
6.368.2.5	toString	1939
6.368.2.6	visit	1939
6.368.3	Field Documentation	1939
6.368.3.1	ID_FLUSHCOMMAND	1939
6.369	activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller Class Reference	1940
6.369.1	Detailed Description	1940
6.369.2	Constructor & Destructor Documentation	1941
6.369.2.1	FlushCommandMarshaller	1941
6.369.2.2	~FlushCommandMarshaller	1941
6.369.3	Member Function Documentation	1941

6.369.3.1 createObject . . . . .	1941
6.369.3.2 getDataStructureType . . . . .	1941
6.369.3.3 looseMarshal . . . . .	1941
6.369.3.4 looseUnmarshal . . . . .	1942
6.369.3.5 tightMarshal1 . . . . .	1942
6.369.3.6 tightMarshal2 . . . . .	1942
6.369.3.7 tightUnmarshal . . . . .	1943
6.370activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference . . . . .	1944
6.370.1 Detailed Description . . . . .	1944
6.370.2 Constructor & Destructor Documentation . . . . .	1945
6.370.2.1 FlushCommandMarshaller . . . . .	1945
6.370.2.2 ~FlushCommandMarshaller . . . . .	1945
6.370.3 Member Function Documentation . . . . .	1945
6.370.3.1 createObject . . . . .	1945
6.370.3.2 getDataStructureType . . . . .	1945
6.370.3.3 looseMarshal . . . . .	1945
6.370.3.4 looseUnmarshal . . . . .	1946
6.370.3.5 tightMarshal1 . . . . .	1946
6.370.3.6 tightMarshal2 . . . . .	1946
6.370.3.7 tightUnmarshal . . . . .	1947
6.371activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference . . . . .	1948
6.371.1 Detailed Description . . . . .	1948
6.371.2 Constructor & Destructor Documentation . . . . .	1949
6.371.2.1 FlushCommandMarshaller . . . . .	1949
6.371.2.2 ~FlushCommandMarshaller . . . . .	1949
6.371.3 Member Function Documentation . . . . .	1949
6.371.3.1 createObject . . . . .	1949
6.371.3.2 getDataStructureType . . . . .	1949
6.371.3.3 looseMarshal . . . . .	1949
6.371.3.4 looseUnmarshal . . . . .	1950
6.371.3.5 tightMarshal1 . . . . .	1950
6.371.3.6 tightMarshal2 . . . . .	1950
6.371.3.7 tightUnmarshal . . . . .	1951
6.372activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference . . . . .	1952



6.372.1 Detailed Description . . . . .	1952
6.372.2 Constructor & Destructor Documentation . . . . .	1953
6.372.2.1 FlushCommandMarshaller . . . . .	1953
6.372.2.2 ~FlushCommandMarshaller . . . . .	1953
6.372.3 Member Function Documentation . . . . .	1953
6.372.3.1 createObject . . . . .	1953
6.372.3.2 getDataStructureType . . . . .	1953
6.372.3.3 looseMarshal . . . . .	1953
6.372.3.4 looseUnmarshal . . . . .	1954
6.372.3.5 tightMarshal1 . . . . .	1954
6.372.3.6 tightMarshal2 . . . . .	1954
6.372.3.7 tightUnmarshal . . . . .	1955
6.373activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference . . . . .	1956
6.373.1 Detailed Description . . . . .	1956
6.373.2 Constructor & Destructor Documentation . . . . .	1957
6.373.2.1 FlushCommandMarshaller . . . . .	1957
6.373.2.2 ~FlushCommandMarshaller . . . . .	1957
6.373.3 Member Function Documentation . . . . .	1957
6.373.3.1 createObject . . . . .	1957
6.373.3.2 getDataStructureType . . . . .	1957
6.373.3.3 looseMarshal . . . . .	1957
6.373.3.4 looseUnmarshal . . . . .	1958
6.373.3.5 tightMarshal1 . . . . .	1958
6.373.3.6 tightMarshal2 . . . . .	1958
6.373.3.7 tightUnmarshal . . . . .	1959
6.374activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference . . . . .	1960
6.374.1 Detailed Description . . . . .	1960
6.374.2 Constructor & Destructor Documentation . . . . .	1961
6.374.2.1 FlushCommandMarshaller . . . . .	1961
6.374.2.2 ~FlushCommandMarshaller . . . . .	1961
6.374.3 Member Function Documentation . . . . .	1961
6.374.3.1 createObject . . . . .	1961
6.374.3.2 getDataStructureType . . . . .	1961
6.374.3.3 looseMarshal . . . . .	1961
6.374.3.4 looseUnmarshal . . . . .	1962

6.374.3.5 tightMarshal1 . . . . .	1962
6.374.3.6 tightMarshal2 . . . . .	1962
6.374.3.7 tightUnmarshal . . . . .	1963
6.375decaf::util::logging::Formatter Class Reference . . . . .	1964
6.375.1 Detailed Description . . . . .	1964
6.375.2 Constructor & Destructor Documentation . . . . .	1964
6.375.2.1 ~Formatter . . . . .	1964
6.375.3 Member Function Documentation . . . . .	1964
6.375.3.1 format . . . . .	1964
6.375.3.2 formatMessage . . . . .	1965
6.375.3.3 getHead . . . . .	1965
6.375.3.4 getTail . . . . .	1965
6.376decaf::util::concurrent::Future< V > Class Template Reference . . . . .	1966
6.376.1 Detailed Description . . . . .	1966
6.376.2 Constructor & Destructor Documentation . . . . .	1967
6.376.2.1 ~Future . . . . .	1967
6.376.3 Member Function Documentation . . . . .	1967
6.376.3.1 cancel . . . . .	1967
6.376.3.2 get . . . . .	1967
6.376.3.3 get . . . . .	1968
6.376.3.4 isCancelled . . . . .	1968
6.376.3.5 isDone . . . . .	1968
6.377activemq::transport::correlator::FutureResponse Class Reference . . . . .	1969
6.377.1 Detailed Description . . . . .	1969
6.377.2 Constructor & Destructor Documentation . . . . .	1969
6.377.2.1 FutureResponse . . . . .	1969
6.377.2.2 ~FutureResponse . . . . .	1969
6.377.3 Member Function Documentation . . . . .	1969
6.377.3.1 getResponse . . . . .	1969
6.377.3.2 getResponse . . . . .	1969
6.377.3.3 getResponse . . . . .	1970
6.377.3.4 getResponse . . . . .	1970
6.377.3.5 setResponse . . . . .	1970
6.378decaf::security::GeneralSecurityException Class Reference . . . . .	1971
6.378.1 Constructor & Destructor Documentation . . . . .	1971
6.378.1.1 GeneralSecurityException . . . . .	1971

6.378.1.2 GeneralSecurityException . . . . .	1971
6.378.1.3 GeneralSecurityException . . . . .	1972
6.378.1.4 GeneralSecurityException . . . . .	1972
6.378.1.5 GeneralSecurityException . . . . .	1972
6.378.1.6 GeneralSecurityException . . . . .	1972
6.378.1.7 ~GeneralSecurityException . . . . .	1973
6.378.2 Member Function Documentation . . . . .	1973
6.378.2.1 clone . . . . .	1973
6.379decaf::internal::util::GenericResource< T > Class Template Reference . . . . .	1974
6.379.1 Detailed Description . . . . .	1974
6.379.2 Constructor & Destructor Documentation . . . . .	1974
6.379.2.1 GenericResource . . . . .	1974
6.379.2.2 ~GenericResource . . . . .	1974
6.379.3 Member Function Documentation . . . . .	1974
6.379.3.1 getManaged . . . . .	1974
6.379.3.2 setManaged . . . . .	1974
6.380gz_header_s Struct Reference . . . . .	1975
6.380.1 Field Documentation . . . . .	1975
6.380.1.1 comm_max . . . . .	1975
6.380.1.2 comment . . . . .	1975
6.380.1.3 done . . . . .	1975
6.380.1.4 extra . . . . .	1975
6.380.1.5 extra_len . . . . .	1975
6.380.1.6 extra_max . . . . .	1975
6.380.1.7 hcrc . . . . .	1975
6.380.1.8 name . . . . .	1975
6.380.1.9 name_max . . . . .	1975
6.380.1.10bs . . . . .	1975
6.380.1.11text . . . . .	1975
6.380.1.12ime . . . . .	1975
6.380.1.13flags . . . . .	1975
6.381gz_state Struct Reference . . . . .	1976
6.381.1 Field Documentation . . . . .	1977
6.381.1.1 direct . . . . .	1977
6.381.1.2 eof . . . . .	1977
6.381.1.3 err . . . . .	1977

6.381.1.4	fd	1977
6.381.1.5	have	1977
6.381.1.6	how	1977
6.381.1.7	in	1977
6.381.1.8	level	1977
6.381.1.9	mode	1977
6.381.1.10	msg	1977
6.381.1.11	next	1977
6.381.1.12	out	1977
6.381.1.13	path	1977
6.381.1.14	pos	1977
6.381.1.15	raw	1977
6.381.1.16	seek	1977
6.381.1.17	size	1977
6.381.1.18	skip	1977
6.381.1.19	start	1977
6.381.1.20	strategy	1977
6.381.1.21	strm	1977
6.381.1.22	want	1977
6.382	decaf::util::logging::Handler Class Reference	1978
6.382.1	Detailed Description	1979
6.382.2	Constructor & Destructor Documentation	1979
6.382.2.1	Handler	1979
6.382.2.2	~Handler	1979
6.382.3	Member Function Documentation	1979
6.382.3.1	flush	1979
6.382.3.2	getErrorManager	1979
6.382.3.3	getFilter	1979
6.382.3.4	getFormatter	1979
6.382.3.5	getLevel	1980
6.382.3.6	isLoggable	1980
6.382.3.7	publish	1980
6.382.3.8	reportError	1980
6.382.3.9	setErrorManager	1980
6.382.3.10	setFilter	1981
6.382.3.11	setFormatter	1981

6.382.3.12	setLevel . . . . .	1981
6.383	decaf::internal::util::HexStringParser Class Reference . . . . .	1982
6.383.1	Constructor & Destructor Documentation . . . . .	1982
6.383.1.1	HexStringParser . . . . .	1982
6.383.1.2	~HexStringParser . . . . .	1982
6.383.2	Member Function Documentation . . . . .	1982
6.383.2.1	parse . . . . .	1982
6.383.2.2	parseDouble . . . . .	1983
6.383.2.3	parseFloat . . . . .	1983
6.384	activemq::wireformat::openwire::utils::HexTable Class Reference . . . . .	1984
6.384.1	Detailed Description . . . . .	1984
6.384.2	Constructor & Destructor Documentation . . . . .	1984
6.384.2.1	HexTable . . . . .	1984
6.384.2.2	~HexTable . . . . .	1984
6.384.3	Member Function Documentation . . . . .	1984
6.384.3.1	operator[] . . . . .	1984
6.384.3.2	operator[] . . . . .	1984
6.384.3.3	size . . . . .	1985
6.385	decaf::net::HttpRetryException Class Reference . . . . .	1986
6.385.1	Constructor & Destructor Documentation . . . . .	1986
6.385.1.1	HttpRetryException . . . . .	1986
6.385.1.2	HttpRetryException . . . . .	1986
6.385.1.3	HttpRetryException . . . . .	1987
6.385.1.4	HttpRetryException . . . . .	1987
6.385.1.5	HttpRetryException . . . . .	1987
6.385.1.6	HttpRetryException . . . . .	1987
6.385.1.7	~HttpRetryException . . . . .	1988
6.385.2	Member Function Documentation . . . . .	1988
6.385.2.1	clone . . . . .	1988
6.386	activemq::util::IdGenerator Class Reference . . . . .	1989
6.386.1	Constructor & Destructor Documentation . . . . .	1989
6.386.1.1	IdGenerator . . . . .	1989
6.386.1.2	IdGenerator . . . . .	1989
6.386.1.3	~IdGenerator . . . . .	1989
6.386.2	Member Function Documentation . . . . .	1989
6.386.2.1	compare . . . . .	1989

6.386.2.2 generateId . . . . .	1990
6.386.2.3 getHostname . . . . .	1990
6.386.2.4 getSeedFromId . . . . .	1990
6.386.2.5 getSequenceFromId . . . . .	1990
6.387decaf::lang::exceptions::IllegalArgumentException Class Reference . . . . .	1991
6.387.1 Constructor & Destructor Documentation . . . . .	1991
6.387.1.1 IllegalArgumentException . . . . .	1991
6.387.1.2 IllegalArgumentException . . . . .	1991
6.387.1.3 IllegalArgumentException . . . . .	1992
6.387.1.4 IllegalArgumentException . . . . .	1992
6.387.1.5 IllegalArgumentException . . . . .	1992
6.387.1.6 IllegalArgumentException . . . . .	1992
6.387.1.7 ~IllegalArgumentException . . . . .	1993
6.387.2 Member Function Documentation . . . . .	1993
6.387.2.1 clone . . . . .	1993
6.388decaf::lang::exceptions::IllegalMonitorStateException Class Reference . . . . .	1994
6.388.1 Constructor & Destructor Documentation . . . . .	1994
6.388.1.1 IllegalMonitorStateException . . . . .	1994
6.388.1.2 IllegalMonitorStateException . . . . .	1994
6.388.1.3 IllegalMonitorStateException . . . . .	1995
6.388.1.4 IllegalMonitorStateException . . . . .	1995
6.388.1.5 IllegalMonitorStateException . . . . .	1995
6.388.1.6 IllegalMonitorStateException . . . . .	1995
6.388.1.7 ~IllegalMonitorStateException . . . . .	1996
6.388.2 Member Function Documentation . . . . .	1996
6.388.2.1 clone . . . . .	1996
6.389cms::IllegalStateException Class Reference . . . . .	1997
6.389.1 Detailed Description . . . . .	1997
6.389.2 Constructor & Destructor Documentation . . . . .	1997
6.389.2.1 IllegalStateException . . . . .	1997
6.389.2.2 IllegalStateException . . . . .	1997
6.389.2.3 IllegalStateException . . . . .	1997
6.389.2.4 IllegalStateException . . . . .	1997
6.389.2.5 ~IllegalStateException . . . . .	1997
6.390decaf::lang::exceptions::IllegalStateException Class Reference . . . . .	1998
6.390.1 Constructor & Destructor Documentation . . . . .	1998

6.390.1.1	IllegalStateException	1998
6.390.1.2	IllegalStateException	1998
6.390.1.3	IllegalStateException	1999
6.390.1.4	IllegalStateException	1999
6.390.1.5	IllegalStateException	1999
6.390.1.6	IllegalStateException	1999
6.390.1.7	~IllegalStateException	2000
6.390.2	Member Function Documentation	2000
6.390.2.1	clone	2000
6.391	decaf::lang::exceptions::IllegalThreadStateException Class Reference	2001
6.391.1	Constructor & Destructor Documentation	2001
6.391.1.1	IllegalThreadStateException	2001
6.391.1.2	IllegalThreadStateException	2001
6.391.1.3	IllegalThreadStateException	2002
6.391.1.4	IllegalThreadStateException	2002
6.391.1.5	IllegalThreadStateException	2002
6.391.1.6	IllegalThreadStateException	2002
6.391.1.7	~IllegalThreadStateException	2003
6.391.2	Member Function Documentation	2003
6.391.2.1	clone	2003
6.392	activemq::transport::inactivity::InactivityMonitor Class Reference	2004
6.392.1	Constructor & Destructor Documentation	2005
6.392.1.1	InactivityMonitor	2005
6.392.1.2	InactivityMonitor	2005
6.392.1.3	~InactivityMonitor	2005
6.392.2	Member Function Documentation	2005
6.392.2.1	close	2005
6.392.2.2	getInitialDelayTime	2005
6.392.2.3	getReadCheckTime	2005
6.392.2.4	getWriteCheckTime	2005
6.392.2.5	isKeepAliveResponseRequired	2005
6.392.2.6	onCommand	2005
6.392.2.7	oneway	2006
6.392.2.8	onException	2006
6.392.2.9	setInitialDelayTime	2007
6.392.2.10	setKeepAliveResponseRequired	2007

6.392.2.1	setReadCheckTime . . . . .	2007
6.392.2.2	setWriteCheckTime . . . . .	2007
6.392.3	Friends And Related Function Documentation . . . . .	2007
6.392.3.1	AsyncSignalReadErrorTask . . . . .	2007
6.392.3.2	AsyncWriteTask . . . . .	2007
6.392.3.3	ReadChecker . . . . .	2007
6.392.3.4	WriteChecker . . . . .	2007
6.393	decaf::lang::exceptions::IndexOutOfBoundsException Class Reference . . . . .	2008
6.393.1	Constructor & Destructor Documentation . . . . .	2008
6.393.1.1	IndexOutOfBoundsException . . . . .	2008
6.393.1.2	IndexOutOfBoundsException . . . . .	2008
6.393.1.3	IndexOutOfBoundsException . . . . .	2009
6.393.1.4	IndexOutOfBoundsException . . . . .	2009
6.393.1.5	IndexOutOfBoundsException . . . . .	2009
6.393.1.6	IndexOutOfBoundsException . . . . .	2009
6.393.1.7	~IndexOutOfBoundsException . . . . .	2010
6.393.2	Member Function Documentation . . . . .	2010
6.393.2.1	clone . . . . .	2010
6.394	decaf::net::Inet4Address Class Reference . . . . .	2011
6.394.1	Constructor & Destructor Documentation . . . . .	2012
6.394.1.1	Inet4Address . . . . .	2012
6.394.1.2	Inet4Address . . . . .	2012
6.394.1.3	Inet4Address . . . . .	2012
6.394.1.4	~Inet4Address . . . . .	2012
6.394.2	Member Function Documentation . . . . .	2012
6.394.2.1	isAnyLocalAddress . . . . .	2012
6.394.2.2	isLinkLocalAddress . . . . .	2012
6.394.2.3	isLoopbackAddress . . . . .	2012
6.394.2.4	isMCGlobal . . . . .	2013
6.394.2.5	isMCLinkLocal . . . . .	2013
6.394.2.6	isMCNodeLocal . . . . .	2013
6.394.2.7	isMCOrgLocal . . . . .	2013
6.394.2.8	isMCSiteLocal . . . . .	2013
6.394.2.9	isMulticastAddress . . . . .	2014
6.394.2.10	sSiteLocalAddress . . . . .	2014
6.394.3	Friends And Related Function Documentation . . . . .	2014



6.394.3.1 InetAddress . . . . .	2014
6.395decaf::net::Inet6Address Class Reference . . . . .	2015
6.395.1 Constructor & Destructor Documentation . . . . .	2015
6.395.1.1 Inet6Address . . . . .	2015
6.395.1.2 Inet6Address . . . . .	2015
6.395.1.3 Inet6Address . . . . .	2015
6.395.1.4 ~Inet6Address . . . . .	2015
6.395.2 Friends And Related Function Documentation . . . . .	2015
6.395.2.1 InetAddress . . . . .	2015
6.396decaf::net::InetAddress Class Reference . . . . .	2016
6.396.1 Detailed Description . . . . .	2018
6.396.2 Constructor & Destructor Documentation . . . . .	2018
6.396.2.1 InetAddress . . . . .	2018
6.396.2.2 InetAddress . . . . .	2018
6.396.2.3 InetAddress . . . . .	2018
6.396.2.4 ~InetAddress . . . . .	2018
6.396.3 Member Function Documentation . . . . .	2018
6.396.3.1 bytesToInt . . . . .	2018
6.396.3.2 getAddress . . . . .	2018
6.396.3.3 getAnyAddress . . . . .	2019
6.396.3.4 getByAddress . . . . .	2019
6.396.3.5 getByAddress . . . . .	2019
6.396.3.6 getHostAddress . . . . .	2019
6.396.3.7 getHostName . . . . .	2019
6.396.3.8 getLocalHost . . . . .	2020
6.396.3.9 getLoopbackAddress . . . . .	2020
6.396.3.10sAnyLocalAddress . . . . .	2020
6.396.3.11sLinkLocalAddress . . . . .	2020
6.396.3.12sLoopbackAddress . . . . .	2020
6.396.3.13sMCGlobal . . . . .	2021
6.396.3.14sMCLinkLocal . . . . .	2021
6.396.3.15sMCNodeLocal . . . . .	2021
6.396.3.16sMCOrgLocal . . . . .	2021
6.396.3.17sMCSiteLocal . . . . .	2021
6.396.3.18sMulticastAddress . . . . .	2022
6.396.3.19sSiteLocalAddress . . . . .	2022

6.396.3.20oString . . . . .	2022
6.396.4Field Documentation . . . . .	2022
6.396.4.1 addressBytes . . . . .	2022
6.396.4.2 anyBytes . . . . .	2022
6.396.4.3 hostname . . . . .	2022
6.396.4.4 loopbackBytes . . . . .	2022
6.396.4.5 reached . . . . .	2022
6.397decaf::net::InetSocketAddress Class Reference . . . . .	2023
6.397.1 Constructor & Destructor Documentation . . . . .	2023
6.397.1.1 InetSocketAddress . . . . .	2023
6.397.1.2 ~InetSocketAddress . . . . .	2023
6.398inflate_state Struct Reference . . . . .	2024
6.398.1 Field Documentation . . . . .	2025
6.398.1.1 back . . . . .	2025
6.398.1.2 bits . . . . .	2025
6.398.1.3 check . . . . .	2025
6.398.1.4 codes . . . . .	2025
6.398.1.5 distbits . . . . .	2025
6.398.1.6 distcode . . . . .	2025
6.398.1.7 dmax . . . . .	2025
6.398.1.8 extra . . . . .	2025
6.398.1.9 flags . . . . .	2025
6.398.1.10have . . . . .	2025
6.398.1.11havedict . . . . .	2025
6.398.1.12head . . . . .	2025
6.398.1.13hold . . . . .	2025
6.398.1.14last . . . . .	2025
6.398.1.15lenbits . . . . .	2025
6.398.1.16encode . . . . .	2025
6.398.1.17length . . . . .	2025
6.398.1.18ens . . . . .	2025
6.398.1.19mode . . . . .	2025
6.398.1.20hcode . . . . .	2025
6.398.1.21ndist . . . . .	2025
6.398.1.22next . . . . .	2025
6.398.1.23hlen . . . . .	2025

6.398.1.24	offset . . . . .	2025
6.398.1.25	same . . . . .	2025
6.398.1.26	total . . . . .	2025
6.398.1.27	was . . . . .	2025
6.398.1.28	vbits . . . . .	2025
6.398.1.29	whave . . . . .	2025
6.398.1.30	window . . . . .	2025
6.398.1.31	wnext . . . . .	2025
6.398.1.32	work . . . . .	2025
6.398.1.33	wrap . . . . .	2025
6.398.1.34	wsz . . . . .	2025
6.399	decaf::util::zip::Inflater Class Reference . . . . .	2027
6.399.1	Detailed Description . . . . .	2028
6.399.2	Constructor & Destructor Documentation . . . . .	2029
6.399.2.1	Inflater . . . . .	2029
6.399.2.2	Inflater . . . . .	2029
6.399.2.3	~Inflater . . . . .	2029
6.399.3	Member Function Documentation . . . . .	2029
6.399.3.1	end . . . . .	2029
6.399.3.2	finish . . . . .	2029
6.399.3.3	finished . . . . .	2029
6.399.3.4	getAdler . . . . .	2029
6.399.3.5	getBytesRead . . . . .	2030
6.399.3.6	getBytesWritten . . . . .	2030
6.399.3.7	getRemaining . . . . .	2030
6.399.3.8	inflate . . . . .	2030
6.399.3.9	inflate . . . . .	2031
6.399.3.10	inflate . . . . .	2031
6.399.3.11	needsDictionary . . . . .	2032
6.399.3.12	needsInput . . . . .	2032
6.399.3.13	reset . . . . .	2032
6.399.3.14	setDictionary . . . . .	2032
6.399.3.15	setDictionary . . . . .	2032
6.399.3.16	setDictionary . . . . .	2033
6.399.3.17	setInput . . . . .	2033
6.399.3.18	setInput . . . . .	2034

6.399.3.19	setInput . . . . .	2034
6.400	decaf::util::zip::InflaterInputStream Class Reference . . . . .	2035
6.400.1	Detailed Description . . . . .	2037
6.400.2	Constructor & Destructor Documentation . . . . .	2037
6.400.2.1	InflaterInputStream . . . . .	2037
6.400.2.2	InflaterInputStream . . . . .	2037
6.400.2.3	InflaterInputStream . . . . .	2038
6.400.2.4	~InflaterInputStream . . . . .	2038
6.400.3	Member Function Documentation . . . . .	2038
6.400.3.1	available . . . . .	2038
6.400.3.2	close . . . . .	2039
6.400.3.3	doReadArrayBounded . . . . .	2039
6.400.3.4	doReadByte . . . . .	2039
6.400.3.5	fill . . . . .	2039
6.400.3.6	mark . . . . .	2039
6.400.3.7	markSupported . . . . .	2040
6.400.3.8	reset . . . . .	2040
6.400.3.9	skip . . . . .	2041
6.400.4	Field Documentation . . . . .	2041
6.400.4.1	atEOF . . . . .	2041
6.400.4.2	buff . . . . .	2041
6.400.4.3	DEFAULT_BUFFER_SIZE . . . . .	2041
6.400.4.4	inflater . . . . .	2041
6.400.4.5	length . . . . .	2041
6.400.4.6	ownInflater . . . . .	2042
6.401	decaf::io::InputStream Class Reference . . . . .	2043
6.401.1	Detailed Description . . . . .	2044
6.401.2	Constructor & Destructor Documentation . . . . .	2045
6.401.2.1	InputStream . . . . .	2045
6.401.2.2	~InputStream . . . . .	2045
6.401.3	Member Function Documentation . . . . .	2045
6.401.3.1	available . . . . .	2045
6.401.3.2	close . . . . .	2045
6.401.3.3	doReadArray . . . . .	2045
6.401.3.4	doReadArrayBounded . . . . .	2046
6.401.3.5	doReadByte . . . . .	2046

6.401.3.6 lock . . . . .	2046
6.401.3.7 mark . . . . .	2046
6.401.3.8 markSupported . . . . .	2047
6.401.3.9 notify . . . . .	2047
6.401.3.10 notifyAll . . . . .	2047
6.401.3.11 read . . . . .	2048
6.401.3.12 read . . . . .	2048
6.401.3.13 read . . . . .	2049
6.401.3.14 reset . . . . .	2049
6.401.3.15 skip . . . . .	2050
6.401.3.16 toString . . . . .	2050
6.401.3.17 tryLock . . . . .	2051
6.401.3.18 unlock . . . . .	2051
6.401.3.19 wait . . . . .	2051
6.401.3.20 wait . . . . .	2052
6.401.3.21 wait . . . . .	2052
6.402 decaf::io::InputStreamReader Class Reference . . . . .	2053
6.402.1 Detailed Description . . . . .	2053
6.402.2 Constructor & Destructor Documentation . . . . .	2054
6.402.2.1 InputStreamReader . . . . .	2054
6.402.2.2 ~InputStreamReader . . . . .	2054
6.402.3 Member Function Documentation . . . . .	2054
6.402.3.1 checkClosed . . . . .	2054
6.402.3.2 close . . . . .	2054
6.402.3.3 doReadArrayBounded . . . . .	2054
6.402.3.4 ready . . . . .	2054
6.403 decaf::internal::nio::IntArrayBuffer Class Reference . . . . .	2056
6.403.1 Constructor & Destructor Documentation . . . . .	2059
6.403.1.1 IntArrayBuffer . . . . .	2059
6.403.1.2 IntArrayBuffer . . . . .	2059
6.403.1.3 IntArrayBuffer . . . . .	2060
6.403.1.4 IntArrayBuffer . . . . .	2060
6.403.1.5 ~IntArrayBuffer . . . . .	2060
6.403.2 Member Function Documentation . . . . .	2060
6.403.2.1 array . . . . .	2060
6.403.2.2 arrayOffset . . . . .	2061

6.403.2.3	asReadOnlyBuffer . . . . .	2061
6.403.2.4	compact . . . . .	2062
6.403.2.5	duplicate . . . . .	2062
6.403.2.6	get . . . . .	2062
6.403.2.7	get . . . . .	2063
6.403.2.8	hasArray . . . . .	2063
6.403.2.9	isReadOnly . . . . .	2063
6.403.2.10	put . . . . .	2063
6.403.2.11	put . . . . .	2064
6.403.2.12	setReadOnly . . . . .	2064
6.403.2.13	slice . . . . .	2064
6.404	decaf::nio::IntBuffer Class Reference . . . . .	2066
6.404.1	Detailed Description . . . . .	2068
6.404.2	Constructor & Destructor Documentation . . . . .	2068
6.404.2.1	IntBuffer . . . . .	2068
6.404.2.2	~IntBuffer . . . . .	2068
6.404.3	Member Function Documentation . . . . .	2068
6.404.3.1	allocate . . . . .	2068
6.404.3.2	array . . . . .	2069
6.404.3.3	arrayOffset . . . . .	2069
6.404.3.4	asReadOnlyBuffer . . . . .	2069
6.404.3.5	compact . . . . .	2070
6.404.3.6	compareTo . . . . .	2070
6.404.3.7	duplicate . . . . .	2070
6.404.3.8	equals . . . . .	2070
6.404.3.9	get . . . . .	2070
6.404.3.10	get . . . . .	2071
6.404.3.11	get . . . . .	2071
6.404.3.12	get . . . . .	2072
6.404.3.13	hasArray . . . . .	2072
6.404.3.14	operator< . . . . .	2072
6.404.3.15	operator== . . . . .	2072
6.404.3.16	put . . . . .	2072
6.404.3.17	put . . . . .	2073
6.404.3.18	put . . . . .	2073
6.404.3.19	put . . . . .	2074

6.404.3.20put . . . . .	2074
6.404.3.21slice . . . . .	2075
6.404.3.22toString . . . . .	2075
6.404.3.23wrap . . . . .	2075
6.404.3.24wrap . . . . .	2075
6.405decaf::lang::Integer Class Reference . . . . .	2077
6.405.1 Constructor & Destructor Documentation . . . . .	2079
6.405.1.1 Integer . . . . .	2079
6.405.1.2 Integer . . . . .	2080
6.405.1.3 ~Integer . . . . .	2080
6.405.2 Member Function Documentation . . . . .	2080
6.405.2.1 bitCount . . . . .	2080
6.405.2.2 byteValue . . . . .	2080
6.405.2.3 compareTo . . . . .	2080
6.405.2.4 compareTo . . . . .	2081
6.405.2.5 decode . . . . .	2081
6.405.2.6 doubleValue . . . . .	2081
6.405.2.7 equals . . . . .	2081
6.405.2.8 equals . . . . .	2082
6.405.2.9 float Value . . . . .	2082
6.405.2.10highestOneBit . . . . .	2082
6.405.2.11int Value . . . . .	2082
6.405.2.12long Value . . . . .	2083
6.405.2.13lowestOneBit . . . . .	2083
6.405.2.14numberOfLeadingZeros . . . . .	2083
6.405.2.15numberOfTrailingZeros . . . . .	2083
6.405.2.16operator< . . . . .	2084
6.405.2.17operator< . . . . .	2084
6.405.2.18operator== . . . . .	2084
6.405.2.19operator== . . . . .	2085
6.405.2.20parseInt . . . . .	2085
6.405.2.21parseInt . . . . .	2085
6.405.2.22reverse . . . . .	2086
6.405.2.23reverseBytes . . . . .	2086
6.405.2.24rotateLeft . . . . .	2086
6.405.2.25rotateRight . . . . .	2087

6.405.2.26	shortValue . . . . .	2087
6.405.2.27	signum . . . . .	2087
6.405.2.28	toBinaryString . . . . .	2087
6.405.2.29	toHexString . . . . .	2088
6.405.2.30	toOctalString . . . . .	2088
6.405.2.31	toString . . . . .	2088
6.405.2.32	toString . . . . .	2089
6.405.2.33	toString . . . . .	2089
6.405.2.34	valueOf . . . . .	2089
6.405.2.35	valueOf . . . . .	2090
6.405.2.36	valueOf . . . . .	2090
6.405.3	Field Documentation . . . . .	2090
6.405.3.1	MAX_VALUE . . . . .	2090
6.405.3.2	MIN_VALUE . . . . .	2090
6.405.3.3	SIZE . . . . .	2090
6.406	activemq::commands::IntegerResponse Class Reference . . . . .	2091
6.406.1	Constructor & Destructor Documentation . . . . .	2092
6.406.1.1	IntegerResponse . . . . .	2092
6.406.1.2	~IntegerResponse . . . . .	2092
6.406.2	Member Function Documentation . . . . .	2092
6.406.2.1	cloneDataStructure . . . . .	2092
6.406.2.2	copyDataStructure . . . . .	2092
6.406.2.3	equals . . . . .	2092
6.406.2.4	getDataStructureType . . . . .	2092
6.406.2.5	getResult . . . . .	2093
6.406.2.6	setResult . . . . .	2093
6.406.2.7	toString . . . . .	2093
6.406.3	Field Documentation . . . . .	2093
6.406.3.1	ID_INTEGERRESPONSE . . . . .	2093
6.406.3.2	result . . . . .	2093
6.407	activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller Class Reference . . . . .	2094
6.407.1	Detailed Description . . . . .	2094
6.407.2	Constructor & Destructor Documentation . . . . .	2095
6.407.2.1	IntegerResponseMarshaller . . . . .	2095
6.407.2.2	~IntegerResponseMarshaller . . . . .	2095
6.407.3	Member Function Documentation . . . . .	2095



6.407.3.1	createObject	2095
6.407.3.2	getDataStructureType	2095
6.407.3.3	looseMarshal	2095
6.407.3.4	looseUnmarshal	2096
6.407.3.5	tightMarshal1	2096
6.407.3.6	tightMarshal2	2097
6.407.3.7	tightUnmarshal	2097
6.408	activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	Class
	Reference	2098
6.408.1	Detailed Description	2098
6.408.2	Constructor & Destructor Documentation	2099
6.408.2.1	IntegerResponseMarshaller	2099
6.408.2.2	~IntegerResponseMarshaller	2099
6.408.3	Member Function Documentation	2099
6.408.3.1	createObject	2099
6.408.3.2	getDataStructureType	2099
6.408.3.3	looseMarshal	2099
6.408.3.4	looseUnmarshal	2100
6.408.3.5	tightMarshal1	2100
6.408.3.6	tightMarshal2	2101
6.408.3.7	tightUnmarshal	2101
6.409	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	Class
	Reference	2102
6.409.1	Detailed Description	2102
6.409.2	Constructor & Destructor Documentation	2103
6.409.2.1	IntegerResponseMarshaller	2103
6.409.2.2	~IntegerResponseMarshaller	2103
6.409.3	Member Function Documentation	2103
6.409.3.1	createObject	2103
6.409.3.2	getDataStructureType	2103
6.409.3.3	looseMarshal	2103
6.409.3.4	looseUnmarshal	2104
6.409.3.5	tightMarshal1	2104
6.409.3.6	tightMarshal2	2105
6.409.3.7	tightUnmarshal	2105
6.410	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	Class
	Reference	2106

6.410.1 Detailed Description . . . . .	2106
6.410.2 Constructor & Destructor Documentation . . . . .	2107
6.410.2.1 IntegerResponseMarshaller . . . . .	2107
6.410.2.2 ~IntegerResponseMarshaller . . . . .	2107
6.410.3 Member Function Documentation . . . . .	2107
6.410.3.1 createObject . . . . .	2107
6.410.3.2 getDataStructureType . . . . .	2107
6.410.3.3 looseMarshal . . . . .	2107
6.410.3.4 looseUnmarshal . . . . .	2108
6.410.3.5 tightMarshal1 . . . . .	2108
6.410.3.6 tightMarshal2 . . . . .	2109
6.410.3.7 tightUnmarshal . . . . .	2109
6.411activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller      Class	
Reference . . . . .	2110
6.411.1 Detailed Description . . . . .	2110
6.411.2 Constructor & Destructor Documentation . . . . .	2111
6.411.2.1 IntegerResponseMarshaller . . . . .	2111
6.411.2.2 ~IntegerResponseMarshaller . . . . .	2111
6.411.3 Member Function Documentation . . . . .	2111
6.411.3.1 createObject . . . . .	2111
6.411.3.2 getDataStructureType . . . . .	2111
6.411.3.3 looseMarshal . . . . .	2111
6.411.3.4 looseUnmarshal . . . . .	2112
6.411.3.5 tightMarshal1 . . . . .	2112
6.411.3.6 tightMarshal2 . . . . .	2113
6.411.3.7 tightUnmarshal . . . . .	2113
6.412activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller      Class	
Reference . . . . .	2114
6.412.1 Detailed Description . . . . .	2114
6.412.2 Constructor & Destructor Documentation . . . . .	2115
6.412.2.1 IntegerResponseMarshaller . . . . .	2115
6.412.2.2 ~IntegerResponseMarshaller . . . . .	2115
6.412.3 Member Function Documentation . . . . .	2115
6.412.3.1 createObject . . . . .	2115
6.412.3.2 getDataStructureType . . . . .	2115
6.412.3.3 looseMarshal . . . . .	2115
6.412.3.4 looseUnmarshal . . . . .	2116

6.412.3.5	tightMarshal	2116
6.412.3.6	tightMarshal2	2117
6.412.3.7	tightUnmarshal	2117
6.413	internal_state Struct Reference	2118
6.413.1	Field Documentation	2120
6.413.1.1	bi_buf	2120
6.413.1.2	bi_valid	2120
6.413.1.3	bl_count	2120
6.413.1.4	bl_desc	2120
6.413.1.5	bl_tree	2120
6.413.1.6	block_start	2120
6.413.1.7	d_buf	2120
6.413.1.8	d_desc	2120
6.413.1.9	depth	2120
6.413.1.10	dummy	2120
6.413.1.11	dyn_dtree	2120
6.413.1.12	dyn_ltree	2120
6.413.1.13	good_match	2120
6.413.1.14	gzhead	2120
6.413.1.15	gzindex	2120
6.413.1.16	hash_bits	2120
6.413.1.17	hash_mask	2120
6.413.1.18	hash_shift	2120
6.413.1.19	hash_size	2120
6.413.1.20	head	2120
6.413.1.21	heap	2120
6.413.1.22	heap_len	2120
6.413.1.23	heap_max	2120
6.413.1.24	high_water	2120
6.413.1.25	ins_h	2120
6.413.1.26	o_buf	2120
6.413.1.27	o_desc	2120
6.413.1.28	oast_eob_len	2120
6.413.1.29	oast_flush	2120
6.413.1.30	oast_lit	2120
6.413.1.31	level	2120

6.413.1.32	lit_bufsize . . . . .	2120
6.413.1.33	lookahead . . . . .	2120
6.413.1.34	match_available . . . . .	2120
6.413.1.35	match_length . . . . .	2120
6.413.1.36	match_start . . . . .	2120
6.413.1.37	matches . . . . .	2120
6.413.1.38	max_chain_length . . . . .	2120
6.413.1.39	max_lazy_match . . . . .	2120
6.413.1.40	method . . . . .	2120
6.413.1.41	nice_match . . . . .	2120
6.413.1.42	opt_len . . . . .	2120
6.413.1.43	pending . . . . .	2120
6.413.1.44	pending_buf . . . . .	2120
6.413.1.45	pending_buf_size . . . . .	2120
6.413.1.46	pending_out . . . . .	2120
6.413.1.47	prev . . . . .	2120
6.413.1.48	prev_length . . . . .	2120
6.413.1.49	prev_match . . . . .	2120
6.413.1.50	static_len . . . . .	2120
6.413.1.51	status . . . . .	2120
6.413.1.52	strategy . . . . .	2120
6.413.1.53	strm . . . . .	2120
6.413.1.54	strstart . . . . .	2120
6.413.1.55	w_bits . . . . .	2120
6.413.1.56	w_mask . . . . .	2120
6.413.1.57	w_size . . . . .	2120
6.413.1.58	window . . . . .	2120
6.413.1.59	window_size . . . . .	2120
6.413.1.60	wrap . . . . .	2120
6.414	activemq::transport::mock::InternalCommandListener Class Reference . . . . .	2122
6.414.1	Detailed Description . . . . .	2122
6.414.2	Constructor & Destructor Documentation . . . . .	2122
6.414.2.1	InternalCommandListener . . . . .	2122
6.414.2.2	~InternalCommandListener . . . . .	2122
6.414.3	Member Function Documentation . . . . .	2122
6.414.3.1	onCommand . . . . .	2122

6.414.3.2 run . . . . .	2123
6.414.3.3 setResponseBuilder . . . . .	2123
6.414.3.4 setTransport . . . . .	2123
6.415decaf::lang::exceptions::InterruptedException Class Reference . . . . .	2124
6.415.1 Constructor & Destructor Documentation . . . . .	2124
6.415.1.1 InterruptedException . . . . .	2124
6.415.1.2 InterruptedException . . . . .	2124
6.415.1.3 InterruptedException . . . . .	2125
6.415.1.4 InterruptedException . . . . .	2125
6.415.1.5 InterruptedException . . . . .	2125
6.415.1.6 InterruptedException . . . . .	2125
6.415.1.7 ~InterruptedException . . . . .	2126
6.415.2 Member Function Documentation . . . . .	2126
6.415.2.1 clone . . . . .	2126
6.416decaf::io::InterruptedException Class Reference . . . . .	2127
6.416.1 Constructor & Destructor Documentation . . . . .	2127
6.416.1.1 InterruptedException . . . . .	2127
6.416.1.2 InterruptedException . . . . .	2127
6.416.1.3 InterruptedException . . . . .	2128
6.416.1.4 InterruptedException . . . . .	2128
6.416.1.5 InterruptedException . . . . .	2128
6.416.1.6 InterruptedException . . . . .	2128
6.416.1.7 ~InterruptedException . . . . .	2129
6.416.2 Member Function Documentation . . . . .	2129
6.416.2.1 clone . . . . .	2129
6.417cms::InvalidClientIdException Class Reference . . . . .	2130
6.417.1 Detailed Description . . . . .	2130
6.417.2 Constructor & Destructor Documentation . . . . .	2130
6.417.2.1 InvalidClientIdException . . . . .	2130
6.417.2.2 InvalidClientIdException . . . . .	2130
6.417.2.3 InvalidClientIdException . . . . .	2130
6.417.2.4 InvalidClientIdException . . . . .	2130
6.417.2.5 ~InvalidClientIdException . . . . .	2130
6.418cms::InvalidDestinationException Class Reference . . . . .	2131
6.418.1 Detailed Description . . . . .	2131
6.418.2 Constructor & Destructor Documentation . . . . .	2131

6.418.2.1 InvalidDestinationException . . . . .	2131
6.418.2.2 InvalidDestinationException . . . . .	2131
6.418.2.3 InvalidDestinationException . . . . .	2131
6.418.2.4 InvalidDestinationException . . . . .	2131
6.418.2.5 ~InvalidDestinationException . . . . .	2131
6.419decaf::security::InvalidKeyException Class Reference . . . . .	2132
6.419.1 Constructor & Destructor Documentation . . . . .	2132
6.419.1.1 InvalidKeyException . . . . .	2132
6.419.1.2 InvalidKeyException . . . . .	2132
6.419.1.3 InvalidKeyException . . . . .	2133
6.419.1.4 InvalidKeyException . . . . .	2133
6.419.1.5 InvalidKeyException . . . . .	2133
6.419.1.6 InvalidKeyException . . . . .	2133
6.419.1.7 ~InvalidKeyException . . . . .	2134
6.419.2 Member Function Documentation . . . . .	2134
6.419.2.1 clone . . . . .	2134
6.420decaf::nio::InvalidMarkException Class Reference . . . . .	2135
6.420.1 Constructor & Destructor Documentation . . . . .	2135
6.420.1.1 InvalidMarkException . . . . .	2135
6.420.1.2 InvalidMarkException . . . . .	2135
6.420.1.3 InvalidMarkException . . . . .	2136
6.420.1.4 InvalidMarkException . . . . .	2136
6.420.1.5 InvalidMarkException . . . . .	2136
6.420.1.6 InvalidMarkException . . . . .	2136
6.420.1.7 ~InvalidMarkException . . . . .	2137
6.420.2 Member Function Documentation . . . . .	2137
6.420.2.1 clone . . . . .	2137
6.421cms::InvalidSelectorException Class Reference . . . . .	2138
6.421.1 Detailed Description . . . . .	2138
6.421.2 Constructor & Destructor Documentation . . . . .	2138
6.421.2.1 InvalidSelectorException . . . . .	2138
6.421.2.2 InvalidSelectorException . . . . .	2138
6.421.2.3 InvalidSelectorException . . . . .	2138
6.421.2.4 InvalidSelectorException . . . . .	2138
6.421.2.5 ~InvalidSelectorException . . . . .	2138
6.422decaf::lang::exceptions::InvalidStateException Class Reference . . . . .	2139

6.422.1 Constructor & Destructor Documentation . . . . .	2139
6.422.1.1 InvalidStateException . . . . .	2139
6.422.1.2 InvalidStateException . . . . .	2139
6.422.1.3 InvalidStateException . . . . .	2140
6.422.1.4 InvalidStateException . . . . .	2140
6.422.1.5 InvalidStateException . . . . .	2140
6.422.1.6 InvalidStateException . . . . .	2140
6.422.1.7 ~InvalidStateException . . . . .	2141
6.422.2 Member Function Documentation . . . . .	2141
6.422.2.1 clone . . . . .	2141
6.423decaf::io::IOException Class Reference . . . . .	2142
6.423.1 Constructor & Destructor Documentation . . . . .	2142
6.423.1.1 IOException . . . . .	2142
6.423.1.2 IOException . . . . .	2142
6.423.1.3 IOException . . . . .	2143
6.423.1.4 IOException . . . . .	2143
6.423.1.5 IOException . . . . .	2143
6.423.1.6 IOException . . . . .	2143
6.423.1.7 ~IOException . . . . .	2144
6.423.2 Member Function Documentation . . . . .	2144
6.423.2.1 clone . . . . .	2144
6.424activemq::transport::IOTransport Class Reference . . . . .	2145
6.424.1 Detailed Description . . . . .	2146
6.424.2 Constructor & Destructor Documentation . . . . .	2146
6.424.2.1 IOTransport . . . . .	2146
6.424.2.2 IOTransport . . . . .	2147
6.424.2.3 ~IOTransport . . . . .	2147
6.424.3 Member Function Documentation . . . . .	2147
6.424.3.1 close . . . . .	2147
6.424.3.2 getRemoteAddress . . . . .	2147
6.424.3.3 getTransportListener . . . . .	2147
6.424.3.4 isClosed . . . . .	2147
6.424.3.5 isConnected . . . . .	2148
6.424.3.6 isFaultTolerant . . . . .	2148
6.424.3.7 narrow . . . . .	2148
6.424.3.8 oneway . . . . .	2148

6.424.3.9 reconnect . . . . .	2149
6.424.3.10 request . . . . .	2149
6.424.3.11 request . . . . .	2149
6.424.3.12 run . . . . .	2150
6.424.3.13 setInputStream . . . . .	2150
6.424.3.14 setOutputStream . . . . .	2150
6.424.3.15 setTransportListener . . . . .	2150
6.424.3.16 setWireFormat . . . . .	2150
6.424.3.17 start . . . . .	2150
6.424.3.18 stop . . . . .	2151
6.425 decaf::lang::Iterable< E > Class Template Reference . . . . .	2152
6.425.1 Detailed Description . . . . .	2152
6.425.2 Constructor & Destructor Documentation . . . . .	2152
6.425.2.1 ~Iterable . . . . .	2152
6.425.3 Member Function Documentation . . . . .	2152
6.425.3.1 iterator . . . . .	2152
6.425.3.2 iterator . . . . .	2152
6.426 decaf::util::Iterator< T > Class Template Reference . . . . .	2154
6.426.1 Detailed Description . . . . .	2154
6.426.2 Constructor & Destructor Documentation . . . . .	2154
6.426.2.1 ~Iterator . . . . .	2154
6.426.3 Member Function Documentation . . . . .	2154
6.426.3.1 hasNext . . . . .	2154
6.426.3.2 next . . . . .	2154
6.426.3.3 remove . . . . .	2155
6.427 activemq::commands::JournalQueueAck Class Reference . . . . .	2156
6.427.1 Constructor & Destructor Documentation . . . . .	2157
6.427.1.1 JournalQueueAck . . . . .	2157
6.427.1.2 ~JournalQueueAck . . . . .	2157
6.427.2 Member Function Documentation . . . . .	2157
6.427.2.1 cloneDataStructure . . . . .	2157
6.427.2.2 copyDataStructure . . . . .	2157
6.427.2.3 equals . . . . .	2157
6.427.2.4 getDataStructureType . . . . .	2157
6.427.2.5 getDestination . . . . .	2158
6.427.2.6 getDestination . . . . .	2158



6.427.2.7	getMessageAck . . . . .	2158
6.427.2.8	getMessageAck . . . . .	2158
6.427.2.9	setDestination . . . . .	2158
6.427.2.10	setMessageAck . . . . .	2158
6.427.2.11	toString . . . . .	2158
6.427.3	Field Documentation . . . . .	2158
6.427.3.1	destination . . . . .	2158
6.427.3.2	ID_JOURNALQUEUEACK . . . . .	2158
6.427.3.3	messageAck . . . . .	2158
6.428	activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller Class	
	Reference . . . . .	2159
6.428.1	Detailed Description . . . . .	2159
6.428.2	Constructor & Destructor Documentation . . . . .	2160
6.428.2.1	JournalQueueAckMarshaller . . . . .	2160
6.428.2.2	~JournalQueueAckMarshaller . . . . .	2160
6.428.3	Member Function Documentation . . . . .	2160
6.428.3.1	createObject . . . . .	2160
6.428.3.2	getDataStructureType . . . . .	2160
6.428.3.3	looseMarshal . . . . .	2160
6.428.3.4	looseUnmarshal . . . . .	2161
6.428.3.5	tightMarshal1 . . . . .	2161
6.428.3.6	tightMarshal2 . . . . .	2161
6.428.3.7	tightUnmarshal . . . . .	2162
6.429	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class	
	Reference . . . . .	2163
6.429.1	Detailed Description . . . . .	2163
6.429.2	Constructor & Destructor Documentation . . . . .	2164
6.429.2.1	JournalQueueAckMarshaller . . . . .	2164
6.429.2.2	~JournalQueueAckMarshaller . . . . .	2164
6.429.3	Member Function Documentation . . . . .	2164
6.429.3.1	createObject . . . . .	2164
6.429.3.2	getDataStructureType . . . . .	2164
6.429.3.3	looseMarshal . . . . .	2164
6.429.3.4	looseUnmarshal . . . . .	2165
6.429.3.5	tightMarshal1 . . . . .	2165
6.429.3.6	tightMarshal2 . . . . .	2165
6.429.3.7	tightUnmarshal . . . . .	2166

6.430	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	Class	
	Reference		2167
6.430.1	Detailed Description		2167
6.430.2	Constructor & Destructor Documentation		2168
6.430.2.1	JournalQueueAckMarshaller		2168
6.430.2.2	~JournalQueueAckMarshaller		2168
6.430.3	Member Function Documentation		2168
6.430.3.1	createObject		2168
6.430.3.2	getDataStructureType		2168
6.430.3.3	looseMarshal		2168
6.430.3.4	looseUnmarshal		2169
6.430.3.5	tightMarshal1		2169
6.430.3.6	tightMarshal2		2169
6.430.3.7	tightUnmarshal		2170
6.431	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	Class	
	Reference		2171
6.431.1	Detailed Description		2171
6.431.2	Constructor & Destructor Documentation		2172
6.431.2.1	JournalQueueAckMarshaller		2172
6.431.2.2	~JournalQueueAckMarshaller		2172
6.431.3	Member Function Documentation		2172
6.431.3.1	createObject		2172
6.431.3.2	getDataStructureType		2172
6.431.3.3	looseMarshal		2172
6.431.3.4	looseUnmarshal		2173
6.431.3.5	tightMarshal1		2173
6.431.3.6	tightMarshal2		2173
6.431.3.7	tightUnmarshal		2174
6.432	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	Class	
	Reference		2175
6.432.1	Detailed Description		2175
6.432.2	Constructor & Destructor Documentation		2176
6.432.2.1	JournalQueueAckMarshaller		2176
6.432.2.2	~JournalQueueAckMarshaller		2176
6.432.3	Member Function Documentation		2176
6.432.3.1	createObject		2176
6.432.3.2	getDataStructureType		2176

6.432.3.3 looseMarshal . . . . .	2176
6.432.3.4 looseUnmarshal . . . . .	2177
6.432.3.5 tightMarshal1 . . . . .	2177
6.432.3.6 tightMarshal2 . . . . .	2177
6.432.3.7 tightUnmarshal . . . . .	2178
6.433activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class Reference . . . . .	2179
6.433.1 Detailed Description . . . . .	2179
6.433.2 Constructor & Destructor Documentation . . . . .	2180
6.433.2.1 JournalQueueAckMarshaller . . . . .	2180
6.433.2.2 ~JournalQueueAckMarshaller . . . . .	2180
6.433.3 Member Function Documentation . . . . .	2180
6.433.3.1 createObject . . . . .	2180
6.433.3.2 getDataStructureType . . . . .	2180
6.433.3.3 looseMarshal . . . . .	2180
6.433.3.4 looseUnmarshal . . . . .	2181
6.433.3.5 tightMarshal1 . . . . .	2181
6.433.3.6 tightMarshal2 . . . . .	2181
6.433.3.7 tightUnmarshal . . . . .	2182
6.434activemq::commands::JournalTopicAck Class Reference . . . . .	2183
6.434.1 Constructor & Destructor Documentation . . . . .	2184
6.434.1.1 JournalTopicAck . . . . .	2184
6.434.1.2 ~JournalTopicAck . . . . .	2184
6.434.2 Member Function Documentation . . . . .	2184
6.434.2.1 cloneDataStructure . . . . .	2184
6.434.2.2 copyDataStructure . . . . .	2184
6.434.2.3 equals . . . . .	2184
6.434.2.4 getClientId . . . . .	2185
6.434.2.5 getClientId . . . . .	2185
6.434.2.6 getDataStructureType . . . . .	2185
6.434.2.7 getDestination . . . . .	2186
6.434.2.8 getDestination . . . . .	2186
6.434.2.9 getMessageId . . . . .	2186
6.434.2.10getMessageId . . . . .	2186
6.434.2.11getMessageSequenceId . . . . .	2186
6.434.2.12getSubscriptionName . . . . .	2186
6.434.2.13getSubscriptionName . . . . .	2186

6.434.2.14	getTransactionId . . . . .	2186
6.434.2.15	getTransactionId . . . . .	2186
6.434.2.16	setClientId . . . . .	2186
6.434.2.17	setDestination . . . . .	2186
6.434.2.18	setMessageId . . . . .	2186
6.434.2.19	setMessageSequenceId . . . . .	2186
6.434.2.20	setSubscriptionName . . . . .	2186
6.434.2.21	setTransactionId . . . . .	2186
6.434.2.22	toString . . . . .	2186
6.434.3	Field Documentation . . . . .	2187
6.434.3.1	clientId . . . . .	2187
6.434.3.2	destination . . . . .	2187
6.434.3.3	ID_ JOURNALTOPICACK . . . . .	2187
6.434.3.4	messageId . . . . .	2187
6.434.3.5	messageSequenceId . . . . .	2187
6.434.3.6	subscriptionName . . . . .	2187
6.434.3.7	transactionId . . . . .	2187
6.435	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class	
	Reference . . . . .	2188
6.435.1	Detailed Description . . . . .	2188
6.435.2	Constructor & Destructor Documentation . . . . .	2189
6.435.2.1	JournalTopicAckMarshaller . . . . .	2189
6.435.2.2	~JournalTopicAckMarshaller . . . . .	2189
6.435.3	Member Function Documentation . . . . .	2189
6.435.3.1	createObject . . . . .	2189
6.435.3.2	getDataStructureType . . . . .	2189
6.435.3.3	looseMarshal . . . . .	2189
6.435.3.4	looseUnmarshal . . . . .	2190
6.435.3.5	tightMarshal1 . . . . .	2190
6.435.3.6	tightMarshal2 . . . . .	2190
6.435.3.7	tightUnmarshal . . . . .	2191
6.436	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller Class	
	Reference . . . . .	2192
6.436.1	Detailed Description . . . . .	2192
6.436.2	Constructor & Destructor Documentation . . . . .	2193
6.436.2.1	JournalTopicAckMarshaller . . . . .	2193
6.436.2.2	~JournalTopicAckMarshaller . . . . .	2193

6.436.3 Member Function Documentation . . . . .	2193
6.436.3.1 createObject . . . . .	2193
6.436.3.2 getDataStructureType . . . . .	2193
6.436.3.3 looseMarshal . . . . .	2193
6.436.3.4 looseUnmarshal . . . . .	2194
6.436.3.5 tightMarshal1 . . . . .	2194
6.436.3.6 tightMarshal2 . . . . .	2194
6.436.3.7 tightUnmarshal . . . . .	2195
6.437activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class	
Reference . . . . .	2196
6.437.1 Detailed Description . . . . .	2196
6.437.2 Constructor & Destructor Documentation . . . . .	2197
6.437.2.1 JournalTopicAckMarshaller . . . . .	2197
6.437.2.2 ~JournalTopicAckMarshaller . . . . .	2197
6.437.3 Member Function Documentation . . . . .	2197
6.437.3.1 createObject . . . . .	2197
6.437.3.2 getDataStructureType . . . . .	2197
6.437.3.3 looseMarshal . . . . .	2197
6.437.3.4 looseUnmarshal . . . . .	2198
6.437.3.5 tightMarshal1 . . . . .	2198
6.437.3.6 tightMarshal2 . . . . .	2198
6.437.3.7 tightUnmarshal . . . . .	2199
6.438activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller Class	
Reference . . . . .	2200
6.438.1 Detailed Description . . . . .	2200
6.438.2 Constructor & Destructor Documentation . . . . .	2201
6.438.2.1 JournalTopicAckMarshaller . . . . .	2201
6.438.2.2 ~JournalTopicAckMarshaller . . . . .	2201
6.438.3 Member Function Documentation . . . . .	2201
6.438.3.1 createObject . . . . .	2201
6.438.3.2 getDataStructureType . . . . .	2201
6.438.3.3 looseMarshal . . . . .	2201
6.438.3.4 looseUnmarshal . . . . .	2202
6.438.3.5 tightMarshal1 . . . . .	2202
6.438.3.6 tightMarshal2 . . . . .	2202
6.438.3.7 tightUnmarshal . . . . .	2203

6.439	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	Class	
	Reference		2204
6.439.1	Detailed Description		2204
6.439.2	Constructor & Destructor Documentation		2205
6.439.2.1	JournalTopicAckMarshaller		2205
6.439.2.2	~JournalTopicAckMarshaller		2205
6.439.3	Member Function Documentation		2205
6.439.3.1	createObject		2205
6.439.3.2	getDataStructureType		2205
6.439.3.3	looseMarshal		2205
6.439.3.4	looseUnmarshal		2206
6.439.3.5	tightMarshal1		2206
6.439.3.6	tightMarshal2		2206
6.439.3.7	tightUnmarshal		2207
6.440	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	Class	
	Reference		2208
6.440.1	Detailed Description		2208
6.440.2	Constructor & Destructor Documentation		2209
6.440.2.1	JournalTopicAckMarshaller		2209
6.440.2.2	~JournalTopicAckMarshaller		2209
6.440.3	Member Function Documentation		2209
6.440.3.1	createObject		2209
6.440.3.2	getDataStructureType		2209
6.440.3.3	looseMarshal		2209
6.440.3.4	looseUnmarshal		2210
6.440.3.5	tightMarshal1		2210
6.440.3.6	tightMarshal2		2210
6.440.3.7	tightUnmarshal		2211
6.441	activemq::commands::JournalTrace	Class Reference	2212
6.441.1	Constructor & Destructor Documentation		2213
6.441.1.1	JournalTrace		2213
6.441.1.2	~JournalTrace		2213
6.441.2	Member Function Documentation		2213
6.441.2.1	cloneDataStructure		2213
6.441.2.2	copyDataStructure		2213
6.441.2.3	equals		2213
6.441.2.4	getDataStructureType		2213

6.441.2.5	getMessage . . . . .	2214
6.441.2.6	getMessage . . . . .	2214
6.441.2.7	setMessage . . . . .	2214
6.441.2.8	toString . . . . .	2214
6.441.3	Field Documentation . . . . .	2214
6.441.3.1	ID_JOURNALTRACE . . . . .	2214
6.441.3.2	message . . . . .	2214
6.442	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference . . . . .	2215
6.442.1	Detailed Description . . . . .	2215
6.442.2	Constructor & Destructor Documentation . . . . .	2216
6.442.2.1	JournalTraceMarshaller . . . . .	2216
6.442.2.2	~JournalTraceMarshaller . . . . .	2216
6.442.3	Member Function Documentation . . . . .	2216
6.442.3.1	createObject . . . . .	2216
6.442.3.2	getDataStructureType . . . . .	2216
6.442.3.3	looseMarshal . . . . .	2216
6.442.3.4	looseUnmarshal . . . . .	2217
6.442.3.5	tightMarshal1 . . . . .	2217
6.442.3.6	tightMarshal2 . . . . .	2217
6.442.3.7	tightUnmarshal . . . . .	2218
6.443	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference . . . . .	2219
6.443.1	Detailed Description . . . . .	2219
6.443.2	Constructor & Destructor Documentation . . . . .	2220
6.443.2.1	JournalTraceMarshaller . . . . .	2220
6.443.2.2	~JournalTraceMarshaller . . . . .	2220
6.443.3	Member Function Documentation . . . . .	2220
6.443.3.1	createObject . . . . .	2220
6.443.3.2	getDataStructureType . . . . .	2220
6.443.3.3	looseMarshal . . . . .	2220
6.443.3.4	looseUnmarshal . . . . .	2221
6.443.3.5	tightMarshal1 . . . . .	2221
6.443.3.6	tightMarshal2 . . . . .	2221
6.443.3.7	tightUnmarshal . . . . .	2222
6.444	activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller Class Reference . . . . .	2223

6.444.1 Detailed Description . . . . .	2223
6.444.2 Constructor & Destructor Documentation . . . . .	2224
6.444.2.1 JournalTraceMarshaller . . . . .	2224
6.444.2.2 ~JournalTraceMarshaller . . . . .	2224
6.444.3 Member Function Documentation . . . . .	2224
6.444.3.1 createObject . . . . .	2224
6.444.3.2 getDataStructureType . . . . .	2224
6.444.3.3 looseMarshal . . . . .	2224
6.444.3.4 looseUnmarshal . . . . .	2225
6.444.3.5 tightMarshal1 . . . . .	2225
6.444.3.6 tightMarshal2 . . . . .	2225
6.444.3.7 tightUnmarshal . . . . .	2226
6.445activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference . . . . .	2227
6.445.1 Detailed Description . . . . .	2227
6.445.2 Constructor & Destructor Documentation . . . . .	2228
6.445.2.1 JournalTraceMarshaller . . . . .	2228
6.445.2.2 ~JournalTraceMarshaller . . . . .	2228
6.445.3 Member Function Documentation . . . . .	2228
6.445.3.1 createObject . . . . .	2228
6.445.3.2 getDataStructureType . . . . .	2228
6.445.3.3 looseMarshal . . . . .	2228
6.445.3.4 looseUnmarshal . . . . .	2229
6.445.3.5 tightMarshal1 . . . . .	2229
6.445.3.6 tightMarshal2 . . . . .	2229
6.445.3.7 tightUnmarshal . . . . .	2230
6.446activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference . . . . .	2231
6.446.1 Detailed Description . . . . .	2231
6.446.2 Constructor & Destructor Documentation . . . . .	2232
6.446.2.1 JournalTraceMarshaller . . . . .	2232
6.446.2.2 ~JournalTraceMarshaller . . . . .	2232
6.446.3 Member Function Documentation . . . . .	2232
6.446.3.1 createObject . . . . .	2232
6.446.3.2 getDataStructureType . . . . .	2232
6.446.3.3 looseMarshal . . . . .	2232
6.446.3.4 looseUnmarshal . . . . .	2233



6.446.3.5 tightMarshal1 . . . . .	2233
6.446.3.6 tightMarshal2 . . . . .	2233
6.446.3.7 tightUnmarshal . . . . .	2234
6.447activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference . . . . .	2235
6.447.1 Detailed Description . . . . .	2235
6.447.2 Constructor & Destructor Documentation . . . . .	2236
6.447.2.1 JournalTraceMarshaller . . . . .	2236
6.447.2.2 ~JournalTraceMarshaller . . . . .	2236
6.447.3 Member Function Documentation . . . . .	2236
6.447.3.1 createObject . . . . .	2236
6.447.3.2 getDataStructureType . . . . .	2236
6.447.3.3 looseMarshal . . . . .	2236
6.447.3.4 looseUnmarshal . . . . .	2237
6.447.3.5 tightMarshal1 . . . . .	2237
6.447.3.6 tightMarshal2 . . . . .	2237
6.447.3.7 tightUnmarshal . . . . .	2238
6.448activemq::commands::JournalTransaction Class Reference . . . . .	2239
6.448.1 Constructor & Destructor Documentation . . . . .	2240
6.448.1.1 JournalTransaction . . . . .	2240
6.448.1.2 ~JournalTransaction . . . . .	2240
6.448.2 Member Function Documentation . . . . .	2240
6.448.2.1 cloneDataStructure . . . . .	2240
6.448.2.2 copyDataStructure . . . . .	2240
6.448.2.3 equals . . . . .	2240
6.448.2.4 getDataStructureType . . . . .	2240
6.448.2.5 getTransactionId . . . . .	2241
6.448.2.6 getTransactionId . . . . .	2241
6.448.2.7 getType . . . . .	2241
6.448.2.8 getWasPrepared . . . . .	2241
6.448.2.9 setTransactionId . . . . .	2241
6.448.2.10 setType . . . . .	2241
6.448.2.11 setWasPrepared . . . . .	2241
6.448.2.12 toString . . . . .	2241
6.448.3 Field Documentation . . . . .	2241
6.448.3.1 ID_JOURNALTRANSACTION . . . . .	2241
6.448.3.2 transactionId . . . . .	2241

6.448.3.3 type . . . . .	2241
6.448.3.4 wasPrepared . . . . .	2241
6.449activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller Class	
Reference . . . . .	2242
6.449.1 Detailed Description . . . . .	2242
6.449.2 Constructor & Destructor Documentation . . . . .	2243
6.449.2.1 JournalTransactionMarshaller . . . . .	2243
6.449.2.2 ~JournalTransactionMarshaller . . . . .	2243
6.449.3 Member Function Documentation . . . . .	2243
6.449.3.1 createObject . . . . .	2243
6.449.3.2 getDataStructureType . . . . .	2243
6.449.3.3 looseMarshal . . . . .	2243
6.449.3.4 looseUnmarshal . . . . .	2244
6.449.3.5 tightMarshal1 . . . . .	2244
6.449.3.6 tightMarshal2 . . . . .	2244
6.449.3.7 tightUnmarshal . . . . .	2245
6.450activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class	
Reference . . . . .	2246
6.450.1 Detailed Description . . . . .	2246
6.450.2 Constructor & Destructor Documentation . . . . .	2247
6.450.2.1 JournalTransactionMarshaller . . . . .	2247
6.450.2.2 ~JournalTransactionMarshaller . . . . .	2247
6.450.3 Member Function Documentation . . . . .	2247
6.450.3.1 createObject . . . . .	2247
6.450.3.2 getDataStructureType . . . . .	2247
6.450.3.3 looseMarshal . . . . .	2247
6.450.3.4 looseUnmarshal . . . . .	2248
6.450.3.5 tightMarshal1 . . . . .	2248
6.450.3.6 tightMarshal2 . . . . .	2248
6.450.3.7 tightUnmarshal . . . . .	2249
6.451activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class	
Reference . . . . .	2250
6.451.1 Detailed Description . . . . .	2250
6.451.2 Constructor & Destructor Documentation . . . . .	2251
6.451.2.1 JournalTransactionMarshaller . . . . .	2251
6.451.2.2 ~JournalTransactionMarshaller . . . . .	2251
6.451.3 Member Function Documentation . . . . .	2251

6.451.3.1 createObject . . . . .	2251
6.451.3.2 getDataStructureType . . . . .	2251
6.451.3.3 looseMarshal . . . . .	2251
6.451.3.4 looseUnmarshal . . . . .	2252
6.451.3.5 tightMarshal1 . . . . .	2252
6.451.3.6 tightMarshal2 . . . . .	2252
6.451.3.7 tightUnmarshal . . . . .	2253
6.452activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class	
Reference . . . . .	2254
6.452.1 Detailed Description . . . . .	2254
6.452.2 Constructor & Destructor Documentation . . . . .	2255
6.452.2.1 JournalTransactionMarshaller . . . . .	2255
6.452.2.2 ~JournalTransactionMarshaller . . . . .	2255
6.452.3 Member Function Documentation . . . . .	2255
6.452.3.1 createObject . . . . .	2255
6.452.3.2 getDataStructureType . . . . .	2255
6.452.3.3 looseMarshal . . . . .	2255
6.452.3.4 looseUnmarshal . . . . .	2256
6.452.3.5 tightMarshal1 . . . . .	2256
6.452.3.6 tightMarshal2 . . . . .	2256
6.452.3.7 tightUnmarshal . . . . .	2257
6.453activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class	
Reference . . . . .	2258
6.453.1 Detailed Description . . . . .	2258
6.453.2 Constructor & Destructor Documentation . . . . .	2259
6.453.2.1 JournalTransactionMarshaller . . . . .	2259
6.453.2.2 ~JournalTransactionMarshaller . . . . .	2259
6.453.3 Member Function Documentation . . . . .	2259
6.453.3.1 createObject . . . . .	2259
6.453.3.2 getDataStructureType . . . . .	2259
6.453.3.3 looseMarshal . . . . .	2259
6.453.3.4 looseUnmarshal . . . . .	2260
6.453.3.5 tightMarshal1 . . . . .	2260
6.453.3.6 tightMarshal2 . . . . .	2260
6.453.3.7 tightUnmarshal . . . . .	2261
6.454activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class	
Reference . . . . .	2262

6.454.1 Detailed Description . . . . .	2262
6.454.2 Constructor & Destructor Documentation . . . . .	2263
6.454.2.1 JournalTransactionMarshaller . . . . .	2263
6.454.2.2 ~JournalTransactionMarshaller . . . . .	2263
6.454.3 Member Function Documentation . . . . .	2263
6.454.3.1 createObject . . . . .	2263
6.454.3.2 getDataStructureType . . . . .	2263
6.454.3.3 looseMarshal . . . . .	2263
6.454.3.4 looseUnmarshal . . . . .	2264
6.454.3.5 tightMarshal1 . . . . .	2264
6.454.3.6 tightMarshal2 . . . . .	2264
6.454.3.7 tightUnmarshal . . . . .	2265
6.455activemq::commands::KeepAliveInfo Class Reference . . . . .	2266
6.455.1 Constructor & Destructor Documentation . . . . .	2267
6.455.1.1 KeepAliveInfo . . . . .	2267
6.455.1.2 ~KeepAliveInfo . . . . .	2267
6.455.2 Member Function Documentation . . . . .	2267
6.455.2.1 cloneDataStructure . . . . .	2267
6.455.2.2 copyDataStructure . . . . .	2267
6.455.2.3 equals . . . . .	2267
6.455.2.4 getDataStructureType . . . . .	2267
6.455.2.5 isKeepAliveInfo . . . . .	2268
6.455.2.6 toString . . . . .	2268
6.455.2.7 visit . . . . .	2268
6.455.3 Field Documentation . . . . .	2268
6.455.3.1 ID_KEEPLIVEINFO . . . . .	2268
6.456activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller Class Refer- ence . . . . .	2269
6.456.1 Detailed Description . . . . .	2269
6.456.2 Constructor & Destructor Documentation . . . . .	2270
6.456.2.1 KeepAliveInfoMarshaller . . . . .	2270
6.456.2.2 ~KeepAliveInfoMarshaller . . . . .	2270
6.456.3 Member Function Documentation . . . . .	2270
6.456.3.1 createObject . . . . .	2270
6.456.3.2 getDataStructureType . . . . .	2270
6.456.3.3 looseMarshal . . . . .	2270
6.456.3.4 looseUnmarshal . . . . .	2271

6.456.3.5 tightMarshal1 . . . . .	2271
6.456.3.6 tightMarshal2 . . . . .	2271
6.456.3.7 tightUnmarshal . . . . .	2272
6.457activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference . . . . .	2273
6.457.1 Detailed Description . . . . .	2273
6.457.2 Constructor & Destructor Documentation . . . . .	2274
6.457.2.1 KeepAliveInfoMarshaller . . . . .	2274
6.457.2.2 ~KeepAliveInfoMarshaller . . . . .	2274
6.457.3 Member Function Documentation . . . . .	2274
6.457.3.1 createObject . . . . .	2274
6.457.3.2 getDataStructureType . . . . .	2274
6.457.3.3 looseMarshal . . . . .	2274
6.457.3.4 looseUnmarshal . . . . .	2275
6.457.3.5 tightMarshal1 . . . . .	2275
6.457.3.6 tightMarshal2 . . . . .	2275
6.457.3.7 tightUnmarshal . . . . .	2276
6.458activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference . . . . .	2277
6.458.1 Detailed Description . . . . .	2277
6.458.2 Constructor & Destructor Documentation . . . . .	2278
6.458.2.1 KeepAliveInfoMarshaller . . . . .	2278
6.458.2.2 ~KeepAliveInfoMarshaller . . . . .	2278
6.458.3 Member Function Documentation . . . . .	2278
6.458.3.1 createObject . . . . .	2278
6.458.3.2 getDataStructureType . . . . .	2278
6.458.3.3 looseMarshal . . . . .	2278
6.458.3.4 looseUnmarshal . . . . .	2279
6.458.3.5 tightMarshal1 . . . . .	2279
6.458.3.6 tightMarshal2 . . . . .	2279
6.458.3.7 tightUnmarshal . . . . .	2280
6.459activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference . . . . .	2281
6.459.1 Detailed Description . . . . .	2281
6.459.2 Constructor & Destructor Documentation . . . . .	2282
6.459.2.1 KeepAliveInfoMarshaller . . . . .	2282
6.459.2.2 ~KeepAliveInfoMarshaller . . . . .	2282

6.459.3 Member Function Documentation . . . . .	2282
6.459.3.1 createObject . . . . .	2282
6.459.3.2 getDataStructureType . . . . .	2282
6.459.3.3 looseMarshal . . . . .	2282
6.459.3.4 looseUnmarshal . . . . .	2283
6.459.3.5 tightMarshal1 . . . . .	2283
6.459.3.6 tightMarshal2 . . . . .	2283
6.459.3.7 tightUnmarshal . . . . .	2284
6.460activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference . . . . .	2285
6.460.1 Detailed Description . . . . .	2285
6.460.2 Constructor & Destructor Documentation . . . . .	2286
6.460.2.1 KeepAliveInfoMarshaller . . . . .	2286
6.460.2.2 ~KeepAliveInfoMarshaller . . . . .	2286
6.460.3 Member Function Documentation . . . . .	2286
6.460.3.1 createObject . . . . .	2286
6.460.3.2 getDataStructureType . . . . .	2286
6.460.3.3 looseMarshal . . . . .	2286
6.460.3.4 looseUnmarshal . . . . .	2287
6.460.3.5 tightMarshal1 . . . . .	2287
6.460.3.6 tightMarshal2 . . . . .	2287
6.460.3.7 tightUnmarshal . . . . .	2288
6.461activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference . . . . .	2289
6.461.1 Detailed Description . . . . .	2289
6.461.2 Constructor & Destructor Documentation . . . . .	2290
6.461.2.1 KeepAliveInfoMarshaller . . . . .	2290
6.461.2.2 ~KeepAliveInfoMarshaller . . . . .	2290
6.461.3 Member Function Documentation . . . . .	2290
6.461.3.1 createObject . . . . .	2290
6.461.3.2 getDataStructureType . . . . .	2290
6.461.3.3 looseMarshal . . . . .	2290
6.461.3.4 looseUnmarshal . . . . .	2291
6.461.3.5 tightMarshal1 . . . . .	2291
6.461.3.6 tightMarshal2 . . . . .	2291
6.461.3.7 tightUnmarshal . . . . .	2292
6.462decaf::security::Key Class Reference . . . . .	2293

6.462.1 Detailed Description . . . . .	2293
6.462.2 Constructor & Destructor Documentation . . . . .	2294
6.462.2.1 ~Key . . . . .	2294
6.462.3 Member Function Documentation . . . . .	2294
6.462.3.1 getAlgorithm . . . . .	2294
6.462.3.2 getEncoded . . . . .	2294
6.462.3.3 getFormat . . . . .	2294
6.463decaf::security::KeyException Class Reference . . . . .	2295
6.463.1 Constructor & Destructor Documentation . . . . .	2295
6.463.1.1 KeyException . . . . .	2295
6.463.1.2 KeyException . . . . .	2295
6.463.1.3 KeyException . . . . .	2296
6.463.1.4 KeyException . . . . .	2296
6.463.1.5 KeyException . . . . .	2296
6.463.1.6 KeyException . . . . .	2296
6.463.1.7 ~KeyException . . . . .	2297
6.463.2 Member Function Documentation . . . . .	2297
6.463.2.1 clone . . . . .	2297
6.464decaf::security::KeyManagementException Class Reference . . . . .	2298
6.464.1 Constructor & Destructor Documentation . . . . .	2298
6.464.1.1 KeyManagementException . . . . .	2298
6.464.1.2 KeyManagementException . . . . .	2298
6.464.1.3 KeyManagementException . . . . .	2299
6.464.1.4 KeyManagementException . . . . .	2299
6.464.1.5 KeyManagementException . . . . .	2299
6.464.1.6 KeyManagementException . . . . .	2299
6.464.1.7 ~KeyManagementException . . . . .	2300
6.464.2 Member Function Documentation . . . . .	2300
6.464.2.1 clone . . . . .	2300
6.465activemq::commands::LastPartialCommand Class Reference . . . . .	2301
6.465.1 Constructor & Destructor Documentation . . . . .	2301
6.465.1.1 LastPartialCommand . . . . .	2301
6.465.1.2 ~LastPartialCommand . . . . .	2301
6.465.2 Member Function Documentation . . . . .	2301
6.465.2.1 cloneDataStructure . . . . .	2301
6.465.2.2 copyDataStructure . . . . .	2302

6.465.2.3 equals . . . . .	2302
6.465.2.4 getDataStructureType . . . . .	2302
6.465.2.5 toString . . . . .	2302
6.465.3 Field Documentation . . . . .	2303
6.465.3.1 ID_LASTPARTIALCOMMAND . . . . .	2303
6.466activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	
Class Reference . . . . .	2304
6.466.1 Detailed Description . . . . .	2304
6.466.2 Constructor & Destructor Documentation . . . . .	2305
6.466.2.1 LastPartialCommandMarshaller . . . . .	2305
6.466.2.2 ~LastPartialCommandMarshaller . . . . .	2305
6.466.3 Member Function Documentation . . . . .	2305
6.466.3.1 createObject . . . . .	2305
6.466.3.2 getDataStructureType . . . . .	2305
6.466.3.3 looseMarshal . . . . .	2305
6.466.3.4 looseUnmarshal . . . . .	2306
6.466.3.5 tightMarshal1 . . . . .	2306
6.466.3.6 tightMarshal2 . . . . .	2307
6.466.3.7 tightUnmarshal . . . . .	2307
6.467activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
Class Reference . . . . .	2308
6.467.1 Detailed Description . . . . .	2308
6.467.2 Constructor & Destructor Documentation . . . . .	2309
6.467.2.1 LastPartialCommandMarshaller . . . . .	2309
6.467.2.2 ~LastPartialCommandMarshaller . . . . .	2309
6.467.3 Member Function Documentation . . . . .	2309
6.467.3.1 createObject . . . . .	2309
6.467.3.2 getDataStructureType . . . . .	2309
6.467.3.3 looseMarshal . . . . .	2309
6.467.3.4 looseUnmarshal . . . . .	2310
6.467.3.5 tightMarshal1 . . . . .	2310
6.467.3.6 tightMarshal2 . . . . .	2311
6.467.3.7 tightUnmarshal . . . . .	2311
6.468activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	
Class Reference . . . . .	2312
6.468.1 Detailed Description . . . . .	2312
6.468.2 Constructor & Destructor Documentation . . . . .	2313



6.468.2.1 LastPartialCommandMarshaller . . . . .	2313
6.468.2.2 ~LastPartialCommandMarshaller . . . . .	2313
6.468.3 Member Function Documentation . . . . .	2313
6.468.3.1 createObject . . . . .	2313
6.468.3.2 getDataStructureType . . . . .	2313
6.468.3.3 looseMarshal . . . . .	2313
6.468.3.4 looseUnmarshal . . . . .	2314
6.468.3.5 tightMarshal1 . . . . .	2314
6.468.3.6 tightMarshal2 . . . . .	2315
6.468.3.7 tightUnmarshal . . . . .	2315
6.469activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
Class Reference . . . . .	2316
6.469.1 Detailed Description . . . . .	2316
6.469.2 Constructor & Destructor Documentation . . . . .	2317
6.469.2.1 LastPartialCommandMarshaller . . . . .	2317
6.469.2.2 ~LastPartialCommandMarshaller . . . . .	2317
6.469.3 Member Function Documentation . . . . .	2317
6.469.3.1 createObject . . . . .	2317
6.469.3.2 getDataStructureType . . . . .	2317
6.469.3.3 looseMarshal . . . . .	2317
6.469.3.4 looseUnmarshal . . . . .	2318
6.469.3.5 tightMarshal1 . . . . .	2318
6.469.3.6 tightMarshal2 . . . . .	2319
6.469.3.7 tightUnmarshal . . . . .	2319
6.470activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
Class Reference . . . . .	2320
6.470.1 Detailed Description . . . . .	2320
6.470.2 Constructor & Destructor Documentation . . . . .	2321
6.470.2.1 LastPartialCommandMarshaller . . . . .	2321
6.470.2.2 ~LastPartialCommandMarshaller . . . . .	2321
6.470.3 Member Function Documentation . . . . .	2321
6.470.3.1 createObject . . . . .	2321
6.470.3.2 getDataStructureType . . . . .	2321
6.470.3.3 looseMarshal . . . . .	2321
6.470.3.4 looseUnmarshal . . . . .	2322
6.470.3.5 tightMarshal1 . . . . .	2322
6.470.3.6 tightMarshal2 . . . . .	2323

6.470.3.7 tightUnmarshal . . . . .	2323
6.471activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
Class Reference . . . . .	2324
6.471.1 Detailed Description . . . . .	2324
6.471.2 Constructor & Destructor Documentation . . . . .	2325
6.471.2.1 LastPartialCommandMarshaller . . . . .	2325
6.471.2.2 ~LastPartialCommandMarshaller . . . . .	2325
6.471.3 Member Function Documentation . . . . .	2325
6.471.3.1 createObject . . . . .	2325
6.471.3.2 getDataStructureType . . . . .	2325
6.471.3.3 looseMarshal . . . . .	2325
6.471.3.4 looseUnmarshal . . . . .	2326
6.471.3.5 tightMarshal1 . . . . .	2326
6.471.3.6 tightMarshal2 . . . . .	2327
6.471.3.7 tightUnmarshal . . . . .	2327
6.472decaf::util::comparators::Less< E > Class Template Reference . . . . .	2328
6.472.1 Detailed Description . . . . .	2328
6.472.2 Constructor & Destructor Documentation . . . . .	2328
6.472.2.1 Less . . . . .	2328
6.472.2.2 ~Less . . . . .	2328
6.472.3 Member Function Documentation . . . . .	2328
6.472.3.1 compare . . . . .	2328
6.472.3.2 operator() . . . . .	2329
6.473std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference . . . . .	2330
6.473.1 Detailed Description . . . . .	2330
6.473.2 Member Function Documentation . . . . .	2330
6.473.2.1 operator() . . . . .	2330
6.474std::less< decaf::lang::Pointer< T > > Struct Template Reference . . . . .	2331
6.474.1 Detailed Description . . . . .	2331
6.474.2 Member Function Documentation . . . . .	2331
6.474.2.1 operator() . . . . .	2331
6.475decaf::util::logging::Level Class Reference . . . . .	2332
6.475.1 Detailed Description . . . . .	2333
6.475.2 Constructor & Destructor Documentation . . . . .	2333
6.475.2.1 Level . . . . .	2333
6.475.2.2 ~Level . . . . .	2334
6.475.3 Member Function Documentation . . . . .	2334

6.475.3.1 compareTo . . . . .	2334
6.475.3.2 equals . . . . .	2334
6.475.3.3 getName . . . . .	2334
6.475.3.4 intValue . . . . .	2334
6.475.3.5 operator< . . . . .	2334
6.475.3.6 operator== . . . . .	2334
6.475.3.7 parse . . . . .	2334
6.475.3.8 toString . . . . .	2335
6.475.4 Field Documentation . . . . .	2335
6.475.4.1 ALL . . . . .	2335
6.475.4.2 CONFIG . . . . .	2335
6.475.4.3 DEBUG . . . . .	2335
6.475.4.4 FINE . . . . .	2335
6.475.4.5 FINER . . . . .	2335
6.475.4.6 FINEST . . . . .	2336
6.475.4.7 INFO . . . . .	2336
6.475.4.8 INHERIT . . . . .	2336
6.475.4.9 OFF . . . . .	2336
6.475.4.10 SEVERE . . . . .	2336
6.475.4.11 WARNING . . . . .	2336
6.476 decaf::util::List< E > Class Template Reference . . . . .	2337
6.476.1 Detailed Description . . . . .	2338
6.476.2 Constructor & Destructor Documentation . . . . .	2338
6.476.2.1 List . . . . .	2338
6.476.2.2 ~List . . . . .	2338
6.476.3 Member Function Documentation . . . . .	2338
6.476.3.1 add . . . . .	2338
6.476.3.2 addAll . . . . .	2339
6.476.3.3 get . . . . .	2339
6.476.3.4 indexOf . . . . .	2340
6.476.3.5 lastIndexOf . . . . .	2340
6.476.3.6 listIterator . . . . .	2341
6.476.3.7 listIterator . . . . .	2341
6.476.3.8 listIterator . . . . .	2341
6.476.3.9 listIterator . . . . .	2342
6.476.3.10 remove . . . . .	2342

6.476.3.1	set . . . . .	2342
6.477	decaf::util::ListIterator< E > Class Template Reference . . . . .	2344
6.477.1	Detailed Description . . . . .	2344
6.477.2	Constructor & Destructor Documentation . . . . .	2345
6.477.2.1	~ListIterator . . . . .	2345
6.477.3	Member Function Documentation . . . . .	2345
6.477.3.1	add . . . . .	2345
6.477.3.2	hasPrevious . . . . .	2345
6.477.3.3	nextIndex . . . . .	2345
6.477.3.4	previous . . . . .	2346
6.477.3.5	previousIndex . . . . .	2346
6.477.3.6	set . . . . .	2346
6.478	activemq::commands::LocalTransactionId Class Reference . . . . .	2347
6.478.1	Member Typedef Documentation . . . . .	2348
6.478.1.1	COMPARATOR . . . . .	2348
6.478.2	Constructor & Destructor Documentation . . . . .	2348
6.478.2.1	LocalTransactionId . . . . .	2348
6.478.2.2	LocalTransactionId . . . . .	2348
6.478.2.3	~LocalTransactionId . . . . .	2348
6.478.3	Member Function Documentation . . . . .	2348
6.478.3.1	cloneDataStructure . . . . .	2348
6.478.3.2	compareTo . . . . .	2348
6.478.3.3	copyDataStructure . . . . .	2348
6.478.3.4	equals . . . . .	2349
6.478.3.5	equals . . . . .	2349
6.478.3.6	getConnectionId . . . . .	2349
6.478.3.7	getConnectionId . . . . .	2349
6.478.3.8	getDataStructureType . . . . .	2349
6.478.3.9	getValue . . . . .	2349
6.478.3.10	operator< . . . . .	2349
6.478.3.11	operator= . . . . .	2349
6.478.3.12	operator== . . . . .	2350
6.478.3.13	setConnectionId . . . . .	2350
6.478.3.14	setValue . . . . .	2350
6.478.3.15	toString . . . . .	2350
6.478.4	Field Documentation . . . . .	2350

6.478.4.1	connectionId	2350
6.478.4.2	ID_LOCALTRANSACTIONID	2350
6.478.4.3	value	2350
6.479	activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller Class	
	Reference	2351
6.479.1	Detailed Description	2351
6.479.2	Constructor & Destructor Documentation	2352
6.479.2.1	LocalTransactionIdMarshaller	2352
6.479.2.2	~LocalTransactionIdMarshaller	2352
6.479.3	Member Function Documentation	2352
6.479.3.1	createObject	2352
6.479.3.2	getDataStructureType	2352
6.479.3.3	looseMarshal	2352
6.479.3.4	looseUnmarshal	2353
6.479.3.5	tightMarshal1	2353
6.479.3.6	tightMarshal2	2353
6.479.3.7	tightUnmarshal	2354
6.480	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class	
	Reference	2355
6.480.1	Detailed Description	2355
6.480.2	Constructor & Destructor Documentation	2356
6.480.2.1	LocalTransactionIdMarshaller	2356
6.480.2.2	~LocalTransactionIdMarshaller	2356
6.480.3	Member Function Documentation	2356
6.480.3.1	createObject	2356
6.480.3.2	getDataStructureType	2356
6.480.3.3	looseMarshal	2356
6.480.3.4	looseUnmarshal	2357
6.480.3.5	tightMarshal1	2357
6.480.3.6	tightMarshal2	2357
6.480.3.7	tightUnmarshal	2358
6.481	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class	
	Reference	2359
6.481.1	Detailed Description	2359
6.481.2	Constructor & Destructor Documentation	2360
6.481.2.1	LocalTransactionIdMarshaller	2360
6.481.2.2	~LocalTransactionIdMarshaller	2360

6.481.3 Member Function Documentation . . . . .	2360
6.481.3.1 createObject . . . . .	2360
6.481.3.2 getDataStructureType . . . . .	2360
6.481.3.3 looseMarshal . . . . .	2360
6.481.3.4 looseUnmarshal . . . . .	2361
6.481.3.5 tightMarshal1 . . . . .	2361
6.481.3.6 tightMarshal2 . . . . .	2361
6.481.3.7 tightUnmarshal . . . . .	2362
6.482activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class	
Reference . . . . .	2363
6.482.1 Detailed Description . . . . .	2363
6.482.2 Constructor & Destructor Documentation . . . . .	2364
6.482.2.1 LocalTransactionIdMarshaller . . . . .	2364
6.482.2.2 ~LocalTransactionIdMarshaller . . . . .	2364
6.482.3 Member Function Documentation . . . . .	2364
6.482.3.1 createObject . . . . .	2364
6.482.3.2 getDataStructureType . . . . .	2364
6.482.3.3 looseMarshal . . . . .	2364
6.482.3.4 looseUnmarshal . . . . .	2365
6.482.3.5 tightMarshal1 . . . . .	2365
6.482.3.6 tightMarshal2 . . . . .	2365
6.482.3.7 tightUnmarshal . . . . .	2366
6.483activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class	
Reference . . . . .	2367
6.483.1 Detailed Description . . . . .	2367
6.483.2 Constructor & Destructor Documentation . . . . .	2368
6.483.2.1 LocalTransactionIdMarshaller . . . . .	2368
6.483.2.2 ~LocalTransactionIdMarshaller . . . . .	2368
6.483.3 Member Function Documentation . . . . .	2368
6.483.3.1 createObject . . . . .	2368
6.483.3.2 getDataStructureType . . . . .	2368
6.483.3.3 looseMarshal . . . . .	2368
6.483.3.4 looseUnmarshal . . . . .	2369
6.483.3.5 tightMarshal1 . . . . .	2369
6.483.3.6 tightMarshal2 . . . . .	2369
6.483.3.7 tightUnmarshal . . . . .	2370

6.484	activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	Class Reference	2371
6.484.1	Detailed Description		2371
6.484.2	Constructor & Destructor Documentation		2372
6.484.2.1	LocalTransactionIdMarshaller		2372
6.484.2.2	~LocalTransactionIdMarshaller		2372
6.484.3	Member Function Documentation		2372
6.484.3.1	createObject		2372
6.484.3.2	getDataStructureType		2372
6.484.3.3	looseMarshal		2372
6.484.3.4	looseUnmarshal		2373
6.484.3.5	tightMarshal1		2373
6.484.3.6	tightMarshal2		2373
6.484.3.7	tightUnmarshal		2374
6.485	decaf::util::concurrent::Lock	Class Reference	2375
6.485.1	Detailed Description		2375
6.485.2	Constructor & Destructor Documentation		2375
6.485.2.1	Lock		2375
6.485.2.2	~Lock		2375
6.485.3	Member Function Documentation		2376
6.485.3.1	isLocked		2376
6.485.3.2	lock		2376
6.485.3.3	unlock		2376
6.486	decaf::util::concurrent::locks::Lock	Class Reference	2377
6.486.1	Detailed Description		2377
6.486.2	Constructor & Destructor Documentation		2378
6.486.2.1	~Lock		2378
6.486.3	Member Function Documentation		2378
6.486.3.1	lock		2378
6.486.3.2	lockInterruptibly		2379
6.486.3.3	newCondition		2379
6.486.3.4	tryLock		2380
6.486.3.5	tryLock		2381
6.486.3.6	unlock		2381
6.487	decaf::util::concurrent::locks::LockSupport	Class Reference	2383
6.487.1	Detailed Description		2383
6.487.2	Constructor & Destructor Documentation		2384

6.487.2.1 ~LockSupport . . . . .	2384
6.487.3 Member Function Documentation . . . . .	2384
6.487.3.1 park . . . . .	2384
6.487.3.2 parkNanos . . . . .	2384
6.487.3.3 parkUntil . . . . .	2385
6.487.3.4 unpark . . . . .	2385
6.488decaf::util::logging::Logger Class Reference . . . . .	2386
6.488.1 Detailed Description . . . . .	2388
6.488.2 Constructor & Destructor Documentation . . . . .	2389
6.488.2.1 Logger . . . . .	2389
6.488.2.2 ~Logger . . . . .	2389
6.488.3 Member Function Documentation . . . . .	2389
6.488.3.1 addHandler . . . . .	2389
6.488.3.2 config . . . . .	2390
6.488.3.3 debug . . . . .	2390
6.488.3.4 entering . . . . .	2390
6.488.3.5 exiting . . . . .	2390
6.488.3.6 fine . . . . .	2391
6.488.3.7 finer . . . . .	2391
6.488.3.8 finest . . . . .	2391
6.488.3.9 getAnonymousLogger . . . . .	2391
6.488.3.10getFilter . . . . .	2392
6.488.3.11getHandlers . . . . .	2392
6.488.3.12getLevel . . . . .	2392
6.488.3.13getLogger . . . . .	2392
6.488.3.14getName . . . . .	2393
6.488.3.15getParent . . . . .	2393
6.488.3.16getUseParentHandlers . . . . .	2393
6.488.3.17info . . . . .	2393
6.488.3.18sLoggable . . . . .	2393
6.488.3.19log . . . . .	2394
6.488.3.20log . . . . .	2394
6.488.3.21log . . . . .	2394
6.488.3.22log . . . . .	2395
6.488.3.23removeHandler . . . . .	2395
6.488.3.24setFilter . . . . .	2395



6.488.3.25	setLevel . . . . .	2395
6.488.3.26	setParent . . . . .	2395
6.488.3.27	setUseParentHandlers . . . . .	2396
6.488.3.28	severe . . . . .	2396
6.488.3.29	throwing . . . . .	2396
6.488.3.30	warning . . . . .	2396
6.489	decaf::util::logging::LoggerHierarchy Class Reference . . . . .	2398
6.489.1	Constructor & Destructor Documentation . . . . .	2398
6.489.1.1	LoggerHierarchy . . . . .	2398
6.489.1.2	~LoggerHierarchy . . . . .	2398
6.490	activemq::io::LoggingInputStream Class Reference . . . . .	2399
6.490.1	Constructor & Destructor Documentation . . . . .	2399
6.490.1.1	LoggingInputStream . . . . .	2399
6.490.1.2	~LoggingInputStream . . . . .	2399
6.490.2	Member Function Documentation . . . . .	2399
6.490.2.1	doReadArrayBounded . . . . .	2399
6.490.2.2	doReadByte . . . . .	2400
6.491	activemq::io::LoggingOutputStream Class Reference . . . . .	2401
6.491.1	Detailed Description . . . . .	2401
6.491.2	Constructor & Destructor Documentation . . . . .	2401
6.491.2.1	LoggingOutputStream . . . . .	2401
6.491.2.2	~LoggingOutputStream . . . . .	2401
6.491.3	Member Function Documentation . . . . .	2401
6.491.3.1	doWriteArrayBounded . . . . .	2401
6.491.3.2	doWriteByte . . . . .	2402
6.492	activemq::transport::logging::LoggingTransport Class Reference . . . . .	2403
6.492.1	Detailed Description . . . . .	2403
6.492.2	Constructor & Destructor Documentation . . . . .	2403
6.492.2.1	LoggingTransport . . . . .	2403
6.492.2.2	~LoggingTransport . . . . .	2404
6.492.3	Member Function Documentation . . . . .	2404
6.492.3.1	onCommand . . . . .	2404
6.492.3.2	oneway . . . . .	2404
6.492.3.3	request . . . . .	2404
6.492.3.4	request . . . . .	2405
6.493	decaf::util::logging::LogManager Class Reference . . . . .	2406

6.493.1 Detailed Description . . . . .	2407
6.493.2 Constructor & Destructor Documentation . . . . .	2408
6.493.2.1 ~LogManager . . . . .	2408
6.493.2.2 LogManager . . . . .	2408
6.493.2.3 LogManager . . . . .	2408
6.493.3 Member Function Documentation . . . . .	2409
6.493.3.1 addLogger . . . . .	2409
6.493.3.2 addPropertyChangeListener . . . . .	2409
6.493.3.3 getLogger . . . . .	2409
6.493.3.4 getLoggerNames . . . . .	2409
6.493.3.5 getLoggerManager . . . . .	2410
6.493.3.6 getProperties . . . . .	2410
6.493.3.7 getProperty . . . . .	2410
6.493.3.8 operator= . . . . .	2410
6.493.3.9 readConfiguration . . . . .	2410
6.493.3.10 readConfiguration . . . . .	2411
6.493.3.11 removePropertyChangeListener . . . . .	2411
6.493.3.12 reset . . . . .	2411
6.493.3.13 setProperties . . . . .	2411
6.493.4 Friends And Related Function Documentation . . . . .	2412
6.493.4.1 decaf::lang::Runtime . . . . .	2412
6.494 decaf::util::logging::LogRecord Class Reference . . . . .	2413
6.494.1 Detailed Description . . . . .	2414
6.494.2 Constructor & Destructor Documentation . . . . .	2414
6.494.2.1 LogRecord . . . . .	2414
6.494.2.2 ~LogRecord . . . . .	2414
6.494.3 Member Function Documentation . . . . .	2414
6.494.3.1 getLevel . . . . .	2414
6.494.3.2 getLoggerName . . . . .	2414
6.494.3.3 getMessage . . . . .	2415
6.494.3.4 getSourceFile . . . . .	2415
6.494.3.5 getSourceFunction . . . . .	2415
6.494.3.6 getSourceLine . . . . .	2415
6.494.3.7 getThrown . . . . .	2415
6.494.3.8 getTimestamp . . . . .	2415
6.494.3.9 getTreadId . . . . .	2416

6.494.3.10	setLevel . . . . .	2416
6.494.3.11	setLoggerName . . . . .	2416
6.494.3.12	setMessage . . . . .	2416
6.494.3.13	setSourceFile . . . . .	2416
6.494.3.14	setSourceFunction . . . . .	2416
6.494.3.15	setSourceLine . . . . .	2417
6.494.3.16	setThrown . . . . .	2417
6.494.3.17	setTimestamp . . . . .	2417
6.494.3.18	setTreadId . . . . .	2417
6.495	decaf::util::logging::LogWriter Class Reference . . . . .	2418
6.495.1	Constructor & Destructor Documentation . . . . .	2418
6.495.1.1	LogWriter . . . . .	2418
6.495.1.2	~LogWriter . . . . .	2418
6.495.2	Member Function Documentation . . . . .	2418
6.495.2.1	destroy . . . . .	2418
6.495.2.2	getInstance . . . . .	2418
6.495.2.3	log . . . . .	2419
6.495.2.4	log . . . . .	2419
6.495.2.5	returnInstance . . . . .	2419
6.496	decaf::lang::Long Class Reference . . . . .	2420
6.496.1	Constructor & Destructor Documentation . . . . .	2422
6.496.1.1	Long . . . . .	2422
6.496.1.2	Long . . . . .	2423
6.496.1.3	~Long . . . . .	2423
6.496.2	Member Function Documentation . . . . .	2423
6.496.2.1	bitCount . . . . .	2423
6.496.2.2	byteValue . . . . .	2423
6.496.2.3	compareTo . . . . .	2423
6.496.2.4	compareTo . . . . .	2424
6.496.2.5	decode . . . . .	2424
6.496.2.6	doubleValue . . . . .	2424
6.496.2.7	equals . . . . .	2424
6.496.2.8	equals . . . . .	2425
6.496.2.9	float Value . . . . .	2425
6.496.2.10	highestOneBit . . . . .	2425
6.496.2.11	int Value . . . . .	2425

6.496.2.12	longValue . . . . .	2426
6.496.2.13	lowestOneBit . . . . .	2426
6.496.2.14	numberOfLeadingZeros . . . . .	2426
6.496.2.15	numberOfTrailingZeros . . . . .	2426
6.496.2.16	operator< . . . . .	2427
6.496.2.17	operator< . . . . .	2427
6.496.2.18	operator== . . . . .	2427
6.496.2.19	operator== . . . . .	2428
6.496.2.20	parseLong . . . . .	2428
6.496.2.21	parseLong . . . . .	2428
6.496.2.22	reverse . . . . .	2429
6.496.2.23	reverseBytes . . . . .	2429
6.496.2.24	rotateLeft . . . . .	2429
6.496.2.25	rotateRight . . . . .	2429
6.496.2.26	shortValue . . . . .	2430
6.496.2.27	signum . . . . .	2430
6.496.2.28	toBinaryString . . . . .	2430
6.496.2.29	toHexString . . . . .	2431
6.496.2.30	toOctalString . . . . .	2431
6.496.2.31	toString . . . . .	2431
6.496.2.32	toString . . . . .	2431
6.496.2.33	toString . . . . .	2432
6.496.2.34	valueOf . . . . .	2432
6.496.2.35	valueOf . . . . .	2432
6.496.2.36	valueOf . . . . .	2432
6.496.3	Field Documentation . . . . .	2433
6.496.3.1	MAX_VALUE . . . . .	2433
6.496.3.2	MIN_VALUE . . . . .	2433
6.496.3.3	SIZE . . . . .	2433
6.497	decaf::internal::nio::LongArrayBuffer Class Reference . . . . .	2434
6.497.1	Constructor & Destructor Documentation . . . . .	2437
6.497.1.1	LongArrayBuffer . . . . .	2437
6.497.1.2	LongArrayBuffer . . . . .	2437
6.497.1.3	LongArrayBuffer . . . . .	2438
6.497.1.4	LongArrayBuffer . . . . .	2438
6.497.1.5	~LongArrayBuffer . . . . .	2438

6.497.2 Member Function Documentation . . . . .	2438
6.497.2.1 array . . . . .	2438
6.497.2.2 arrayOffset . . . . .	2439
6.497.2.3 asReadOnlyBuffer . . . . .	2439
6.497.2.4 compact . . . . .	2440
6.497.2.5 duplicate . . . . .	2440
6.497.2.6 get . . . . .	2440
6.497.2.7 get . . . . .	2441
6.497.2.8 hasArray . . . . .	2441
6.497.2.9 isReadOnly . . . . .	2441
6.497.2.10put . . . . .	2441
6.497.2.11put . . . . .	2442
6.497.2.12setReadOnly . . . . .	2442
6.497.2.13lice . . . . .	2442
6.498decaf::nio::LongBuffer Class Reference . . . . .	2444
6.498.1 Detailed Description . . . . .	2446
6.498.2 Constructor & Destructor Documentation . . . . .	2446
6.498.2.1 LongBuffer . . . . .	2446
6.498.2.2 ~LongBuffer . . . . .	2446
6.498.3 Member Function Documentation . . . . .	2446
6.498.3.1 allocate . . . . .	2446
6.498.3.2 array . . . . .	2447
6.498.3.3 arrayOffset . . . . .	2447
6.498.3.4 asReadOnlyBuffer . . . . .	2447
6.498.3.5 compact . . . . .	2448
6.498.3.6 compareTo . . . . .	2448
6.498.3.7 duplicate . . . . .	2448
6.498.3.8 equals . . . . .	2448
6.498.3.9 get . . . . .	2448
6.498.3.10get . . . . .	2449
6.498.3.11get . . . . .	2449
6.498.3.12get . . . . .	2450
6.498.3.13hasArray . . . . .	2450
6.498.3.14operator< . . . . .	2450
6.498.3.15operator== . . . . .	2450
6.498.3.16put . . . . .	2450

6.498.3.17put . . . . .	2451
6.498.3.18put . . . . .	2451
6.498.3.19put . . . . .	2452
6.498.3.20put . . . . .	2452
6.498.3.21slice . . . . .	2453
6.498.3.22toString . . . . .	2453
6.498.3.23wrap . . . . .	2453
6.498.3.24wrap . . . . .	2454
6.499activemq::util::LongSequenceGenerator Class Reference . . . . .	2455
6.499.1 Detailed Description . . . . .	2455
6.499.2 Constructor & Destructor Documentation . . . . .	2455
6.499.2.1 LongSequenceGenerator . . . . .	2455
6.499.2.2 ~LongSequenceGenerator . . . . .	2455
6.499.3 Member Function Documentation . . . . .	2455
6.499.3.1 getLastSequenceId . . . . .	2455
6.499.3.2 getNextSequenceId . . . . .	2455
6.500decaf::net::MalformedURLException Class Reference . . . . .	2456
6.500.1 Constructor & Destructor Documentation . . . . .	2456
6.500.1.1 MalformedURLException . . . . .	2456
6.500.1.2 MalformedURLException . . . . .	2456
6.500.1.3 MalformedURLException . . . . .	2457
6.500.1.4 MalformedURLException . . . . .	2457
6.500.1.5 MalformedURLException . . . . .	2457
6.500.1.6 MalformedURLException . . . . .	2457
6.500.1.7 ~MalformedURLException . . . . .	2458
6.500.2 Member Function Documentation . . . . .	2458
6.500.2.1 clone . . . . .	2458
6.501decaf::util::Map< K, V, COMPARATOR > Class Template Reference . . . . .	2459
6.501.1 Detailed Description . . . . .	2460
6.501.2 Constructor & Destructor Documentation . . . . .	2460
6.501.2.1 Map . . . . .	2460
6.501.2.2 ~Map . . . . .	2460
6.501.3 Member Function Documentation . . . . .	2460
6.501.3.1 clear . . . . .	2460
6.501.3.2 containsKey . . . . .	2461
6.501.3.3 containsValue . . . . .	2462

6.501.3.4 copy . . . . .	2463
6.501.3.5 equals . . . . .	2463
6.501.3.6 get . . . . .	2463
6.501.3.7 get . . . . .	2464
6.501.3.8 isEmpty . . . . .	2465
6.501.3.9 keySet . . . . .	2466
6.501.3.10put . . . . .	2467
6.501.3.11putAll . . . . .	2468
6.501.3.12remove . . . . .	2468
6.501.3.13size . . . . .	2469
6.501.3.14values . . . . .	2470
6.502cms::MapMessage Class Reference . . . . .	2472
6.502.1 Detailed Description . . . . .	2473
6.502.2 Constructor & Destructor Documentation . . . . .	2474
6.502.2.1 ~MapMessage . . . . .	2474
6.502.3 Member Function Documentation . . . . .	2474
6.502.3.1 getBoolean . . . . .	2474
6.502.3.2 getByte . . . . .	2475
6.502.3.3 getBytes . . . . .	2475
6.502.3.4 getChar . . . . .	2475
6.502.3.5 getDouble . . . . .	2476
6.502.3.6 getFloat . . . . .	2476
6.502.3.7 getInt . . . . .	2476
6.502.3.8 getLong . . . . .	2477
6.502.3.9 getMapNames . . . . .	2477
6.502.3.10getShort . . . . .	2477
6.502.3.11getString . . . . .	2477
6.502.3.12temExists . . . . .	2478
6.502.3.13setBoolean . . . . .	2478
6.502.3.14setByte . . . . .	2478
6.502.3.15setBytes . . . . .	2479
6.502.3.16setChar . . . . .	2479
6.502.3.17setDouble . . . . .	2480
6.502.3.18setFloat . . . . .	2480
6.502.3.19setInt . . . . .	2480
6.502.3.20setLong . . . . .	2481

6.502.3.2	setShort . . . . .	2481
6.502.3.2	setString . . . . .	2481
6.503	decaf::util::logging::MarkBlockLogger Class Reference . . . . .	2483
6.503.1	Detailed Description . . . . .	2483
6.503.2	Constructor & Destructor Documentation . . . . .	2483
6.503.2.1	MarkBlockLogger . . . . .	2483
6.503.2.2	~MarkBlockLogger . . . . .	2483
6.504	activemq::wireformat::MarshalAware Class Reference . . . . .	2484
6.504.1	Constructor & Destructor Documentation . . . . .	2484
6.504.1.1	~MarshalAware . . . . .	2484
6.504.2	Member Function Documentation . . . . .	2484
6.504.2.1	afterMarshal . . . . .	2484
6.504.2.2	afterUnmarshal . . . . .	2485
6.504.2.3	beforeMarshal . . . . .	2485
6.504.2.4	beforeUnmarshal . . . . .	2485
6.504.2.5	getMarshaledForm . . . . .	2485
6.504.2.6	isMarshalAware . . . . .	2486
6.504.2.7	setMarshaledForm . . . . .	2486
6.505	activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference . . . . .	2487
6.505.1	Detailed Description . . . . .	2487
6.505.2	Constructor & Destructor Documentation . . . . .	2487
6.505.2.1	~MarshallerFactory . . . . .	2487
6.505.3	Member Function Documentation . . . . .	2487
6.505.3.1	configure . . . . .	2487
6.506	activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference . . . . .	2488
6.506.1	Detailed Description . . . . .	2488
6.506.2	Constructor & Destructor Documentation . . . . .	2488
6.506.2.1	~MarshallerFactory . . . . .	2488
6.506.3	Member Function Documentation . . . . .	2488
6.506.3.1	configure . . . . .	2488
6.507	activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference . . . . .	2489
6.507.1	Detailed Description . . . . .	2489
6.507.2	Constructor & Destructor Documentation . . . . .	2489
6.507.2.1	~MarshallerFactory . . . . .	2489
6.507.3	Member Function Documentation . . . . .	2489
6.507.3.1	configure . . . . .	2489



6.508	activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference . .	2490
6.508.1	Detailed Description . . . . .	2490
6.508.2	Constructor & Destructor Documentation . . . . .	2490
6.508.2.1	~MarshallerFactory . . . . .	2490
6.508.3	Member Function Documentation . . . . .	2490
6.508.3.1	configure . . . . .	2490
6.509	activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference . .	2491
6.509.1	Detailed Description . . . . .	2491
6.509.2	Constructor & Destructor Documentation . . . . .	2491
6.509.2.1	~MarshallerFactory . . . . .	2491
6.509.3	Member Function Documentation . . . . .	2491
6.509.3.1	configure . . . . .	2491
6.510	activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference . .	2492
6.510.1	Detailed Description . . . . .	2492
6.510.2	Constructor & Destructor Documentation . . . . .	2492
6.510.2.1	~MarshallerFactory . . . . .	2492
6.510.3	Member Function Documentation . . . . .	2492
6.510.3.1	configure . . . . .	2492
6.511	activemq::util::MarshallingSupport Class Reference . . . . .	2493
6.511.1	Constructor & Destructor Documentation . . . . .	2494
6.511.1.1	MarshallingSupport . . . . .	2494
6.511.1.2	~MarshallingSupport . . . . .	2494
6.511.2	Member Function Documentation . . . . .	2494
6.511.2.1	asciiToModifiedUtf8 . . . . .	2494
6.511.2.2	modifiedUtf8ToAscii . . . . .	2494
6.511.2.3	readString16 . . . . .	2495
6.511.2.4	readString32 . . . . .	2495
6.511.2.5	writeString . . . . .	2495
6.511.2.6	writeString16 . . . . .	2496
6.511.2.7	writeString32 . . . . .	2496
6.512	decaf::lang::Math Class Reference . . . . .	2497
6.512.1	Detailed Description . . . . .	2499
6.512.2	Constructor & Destructor Documentation . . . . .	2499
6.512.2.1	Math . . . . .	2499
6.512.2.2	~Math . . . . .	2499
6.512.3	Member Function Documentation . . . . .	2499

6.512.3.1	abs	2499
6.512.3.2	abs	2499
6.512.3.3	abs	2499
6.512.3.4	abs	2500
6.512.3.5	ceil	2500
6.512.3.6	floor	2501
6.512.3.7	max	2501
6.512.3.8	max	2502
6.512.3.9	max	2502
6.512.3.10	max	2502
6.512.3.11	max	2503
6.512.3.12	min	2503
6.512.3.13	min	2503
6.512.3.14	min	2503
6.512.3.15	min	2504
6.512.3.16	min	2504
6.512.3.17	min	2504
6.512.3.18	pow	2505
6.512.3.19	random	2505
6.512.3.20	round	2506
6.512.3.21	round	2506
6.512.3.22	signum	2507
6.512.3.23	signum	2507
6.512.3.24	qrt	2508
6.512.3.25	toDegrees	2511
6.512.3.26	toRadians	2511
6.512.4	Field Documentation	2511
6.512.4.1	E	2511
6.512.4.2	PI	2511
6.513	activemq::util::MemoryUsage Class Reference	2512
6.513.1	Constructor & Destructor Documentation	2513
6.513.1.1	MemoryUsage	2513
6.513.1.2	MemoryUsage	2513
6.513.1.3	~MemoryUsage	2513
6.513.2	Member Function Documentation	2513
6.513.2.1	decreaseUsage	2513

6.513.2.2 enqueueUsage . . . . .	2513
6.513.2.3 getLimit . . . . .	2513
6.513.2.4 getUsage . . . . .	2514
6.513.2.5 increaseUsage . . . . .	2514
6.513.2.6 isFull . . . . .	2514
6.513.2.7 setLimit . . . . .	2514
6.513.2.8 setUsage . . . . .	2514
6.513.2.9 waitForSpace . . . . .	2514
6.513.2.10waitForSpace . . . . .	2515
6.514activemq::commands::Message Class Reference . . . . .	2516
6.514.1 Constructor & Destructor Documentation . . . . .	2520
6.514.1.1 Message . . . . .	2520
6.514.1.2 ~Message . . . . .	2520
6.514.2 Member Function Documentation . . . . .	2520
6.514.2.1 afterUnmarshal . . . . .	2520
6.514.2.2 beforeMarshal . . . . .	2520
6.514.2.3 cloneDataStructure . . . . .	2520
6.514.2.4 copyDataStructure . . . . .	2521
6.514.2.5 equals . . . . .	2521
6.514.2.6 getAckHandler . . . . .	2522
6.514.2.7 getArrival . . . . .	2522
6.514.2.8 getBrokerInTime . . . . .	2522
6.514.2.9 getBrokerOutTime . . . . .	2522
6.514.2.10getBrokerPath . . . . .	2522
6.514.2.11getBrokerPath . . . . .	2522
6.514.2.12getCluster . . . . .	2522
6.514.2.13getCluster . . . . .	2522
6.514.2.14getConnection . . . . .	2522
6.514.2.15getContent . . . . .	2523
6.514.2.16getContent . . . . .	2523
6.514.2.17getCorrelationId . . . . .	2523
6.514.2.18getCorrelationId . . . . .	2523
6.514.2.19getDataStructure . . . . .	2523
6.514.2.20getDataStructure . . . . .	2523
6.514.2.21getDataStructureType . . . . .	2523
6.514.2.22getDestination . . . . .	2524

6.514.2.23	getDestination . . . . .	2524
6.514.2.24	getExpiration . . . . .	2524
6.514.2.25	getGroupID . . . . .	2524
6.514.2.26	getGroupID . . . . .	2524
6.514.2.27	getGroupSequence . . . . .	2524
6.514.2.28	getMarshaledProperties . . . . .	2524
6.514.2.29	getMarshaledProperties . . . . .	2524
6.514.2.30	getMessageId . . . . .	2524
6.514.2.31	getMessageId . . . . .	2524
6.514.2.32	getMessageProperties . . . . .	2524
6.514.2.33	getMessageProperties . . . . .	2524
6.514.2.34	getOriginalDestination . . . . .	2525
6.514.2.35	getOriginalDestination . . . . .	2525
6.514.2.36	getOriginalTransactionId . . . . .	2525
6.514.2.37	getOriginalTransactionId . . . . .	2525
6.514.2.38	getPriority . . . . .	2525
6.514.2.39	getProducerId . . . . .	2525
6.514.2.40	getProducerId . . . . .	2525
6.514.2.41	getRedeliveryCounter . . . . .	2525
6.514.2.42	getReplyTo . . . . .	2525
6.514.2.43	getReplyTo . . . . .	2525
6.514.2.44	getSize . . . . .	2525
6.514.2.45	getTargetConsumerId . . . . .	2526
6.514.2.46	getTargetConsumerId . . . . .	2526
6.514.2.47	getTimestamp . . . . .	2526
6.514.2.48	getTransactionId . . . . .	2526
6.514.2.49	getTransactionId . . . . .	2526
6.514.2.50	getType . . . . .	2526
6.514.2.51	getType . . . . .	2526
6.514.2.52	getUserID . . . . .	2526
6.514.2.53	getUserID . . . . .	2526
6.514.2.54	isCompressed . . . . .	2526
6.514.2.55	isDroppable . . . . .	2526
6.514.2.56	isExpired . . . . .	2526
6.514.2.57	isMarshalAware . . . . .	2526
6.514.2.58	isMessage . . . . .	2527

6.514.2.59	isPersistent	2527
6.514.2.60	isReadOnlyBody	2527
6.514.2.61	isReadOnlyProperties	2527
6.514.2.62	isRecievedByDFBridge	2527
6.514.2.63	isSend	2527
6.514.2.64	setAckHandler	2528
6.514.2.65	setArrival	2528
6.514.2.66	setBrokerInTime	2528
6.514.2.67	setBrokerOutTime	2528
6.514.2.68	setBrokerPath	2528
6.514.2.69	setCluster	2528
6.514.2.70	setCompressed	2528
6.514.2.71	setConnection	2528
6.514.2.72	setContent	2529
6.514.2.73	setCorrelationId	2529
6.514.2.74	setDataStructure	2529
6.514.2.75	setDestination	2529
6.514.2.76	setDroppable	2529
6.514.2.77	setExpiration	2529
6.514.2.78	setGroupID	2529
6.514.2.79	setGroupSequence	2529
6.514.2.80	setMarshallledProperties	2529
6.514.2.81	setMessageId	2529
6.514.2.82	setOriginalDestination	2529
6.514.2.83	setOriginalTransactionId	2529
6.514.2.84	setPersistent	2529
6.514.2.85	setPriority	2529
6.514.2.86	setProducerId	2529
6.514.2.87	setReadOnlyBody	2529
6.514.2.88	setReadOnlyProperties	2530
6.514.2.89	setRecievedByDFBridge	2530
6.514.2.90	setRedeliveryCounter	2530
6.514.2.91	setReplyTo	2530
6.514.2.92	setTargetConsumerId	2530
6.514.2.93	setTimestamp	2530
6.514.2.94	setTransactionId	2530

6.514.2.95	set Type . . . . .	2530
6.514.2.96	set UserID . . . . .	2530
6.514.2.97	toString . . . . .	2530
6.514.2.98	visit . . . . .	2531
6.514.3	Field Documentation . . . . .	2532
6.514.3.1	arrival . . . . .	2532
6.514.3.2	brokerInTime . . . . .	2532
6.514.3.3	brokerOutTime . . . . .	2532
6.514.3.4	brokerPath . . . . .	2532
6.514.3.5	cluster . . . . .	2532
6.514.3.6	compressed . . . . .	2532
6.514.3.7	connection . . . . .	2532
6.514.3.8	content . . . . .	2532
6.514.3.9	correlationId . . . . .	2532
6.514.3.10	dataStructure . . . . .	2532
6.514.3.11	DEFAULT_MESSAGE_SIZE . . . . .	2532
6.514.3.12	destination . . . . .	2532
6.514.3.13	droppable . . . . .	2532
6.514.3.14	expiration . . . . .	2532
6.514.3.15	groupId . . . . .	2532
6.514.3.16	groupSequence . . . . .	2532
6.514.3.17	ID_MESSAGE . . . . .	2532
6.514.3.18	marshalledProperties . . . . .	2532
6.514.3.19	messageId . . . . .	2532
6.514.3.20	originalDestination . . . . .	2532
6.514.3.21	originalTransactionId . . . . .	2532
6.514.3.22	persistent . . . . .	2532
6.514.3.23	priority . . . . .	2532
6.514.3.24	producerId . . . . .	2532
6.514.3.25	recievedByDFBridge . . . . .	2532
6.514.3.26	redeliveryCounter . . . . .	2532
6.514.3.27	replyTo . . . . .	2532
6.514.3.28	targetConsumerId . . . . .	2532
6.514.3.29	timestamp . . . . .	2532
6.514.3.30	transactionId . . . . .	2532
6.514.3.31	type . . . . .	2532

6.514.3.32	userID	2532
6.515	cms::Message Class Reference	2534
6.515.1	Detailed Description	2537
6.515.2	Constructor & Destructor Documentation	2538
6.515.2.1	~Message	2538
6.515.3	Member Function Documentation	2538
6.515.3.1	acknowledge	2538
6.515.3.2	clearBody	2539
6.515.3.3	clearProperties	2539
6.515.3.4	clone	2539
6.515.3.5	getBooleanProperty	2540
6.515.3.6	getByteProperty	2540
6.515.3.7	getCMSCorrelationID	2541
6.515.3.8	getCMSDeliveryMode	2541
6.515.3.9	getCMSDestination	2541
6.515.3.10	getCMSExpiration	2542
6.515.3.11	getCMSMessageID	2542
6.515.3.12	getCMSPriority	2543
6.515.3.13	getCMSRedelivered	2544
6.515.3.14	getCMSReplyTo	2544
6.515.3.15	getCMSTimestamp	2544
6.515.3.16	getCMSType	2545
6.515.3.17	getDoubleProperty	2545
6.515.3.18	getFloatProperty	2546
6.515.3.19	getIntProperty	2546
6.515.3.20	getLongProperty	2547
6.515.3.21	getPropertyNames	2547
6.515.3.22	getShortProperty	2548
6.515.3.23	getStringProperty	2548
6.515.3.24	propertyExists	2549
6.515.3.25	setBooleanProperty	2549
6.515.3.26	setByteProperty	2550
6.515.3.27	setCMSCorrelationID	2550
6.515.3.28	setCMSDeliveryMode	2551
6.515.3.29	setCMSDestination	2551
6.515.3.30	setCMSExpiration	2552

6.515.3.31	setCMSMessageID . . . . .	2552
6.515.3.32	setCMSPriority . . . . .	2552
6.515.3.33	setCMSRedelivered . . . . .	2553
6.515.3.34	setCMSReplyTo . . . . .	2553
6.515.3.35	setCMSTimestamp . . . . .	2554
6.515.3.36	setCMSType . . . . .	2554
6.515.3.37	setDoubleProperty . . . . .	2555
6.515.3.38	setFloatProperty . . . . .	2555
6.515.3.39	setIntProperty . . . . .	2556
6.515.3.40	setLongProperty . . . . .	2556
6.515.3.41	setShortProperty . . . . .	2557
6.515.3.42	setStringProperty . . . . .	2557
6.516	activemq::commands::MessageAck Class Reference . . . . .	2559
6.516.1	Constructor & Destructor Documentation . . . . .	2560
6.516.1.1	MessageAck . . . . .	2560
6.516.1.2	~MessageAck . . . . .	2560
6.516.2	Member Function Documentation . . . . .	2560
6.516.2.1	cloneDataStructure . . . . .	2560
6.516.2.2	copyDataStructure . . . . .	2560
6.516.2.3	equals . . . . .	2560
6.516.2.4	getAckType . . . . .	2561
6.516.2.5	getConsumerId . . . . .	2561
6.516.2.6	getConsumerId . . . . .	2561
6.516.2.7	getDataStructureType . . . . .	2561
6.516.2.8	getDestination . . . . .	2562
6.516.2.9	getDestination . . . . .	2562
6.516.2.10	getFirstMessageId . . . . .	2562
6.516.2.11	getFirstMessageId . . . . .	2562
6.516.2.12	getLastMessageId . . . . .	2562
6.516.2.13	getLastMessageId . . . . .	2562
6.516.2.14	getMessageCount . . . . .	2562
6.516.2.15	getTransactionId . . . . .	2562
6.516.2.16	getTransactionId . . . . .	2562
6.516.2.17	isMessageAck . . . . .	2562
6.516.2.18	setAckType . . . . .	2563
6.516.2.19	setConsumerId . . . . .	2563



6.516.2.20	setDestination . . . . .	2563
6.516.2.21	setFirstMessageId . . . . .	2563
6.516.2.22	setLastMessageId . . . . .	2563
6.516.2.23	setMessageCount . . . . .	2563
6.516.2.24	setTransactionId . . . . .	2563
6.516.2.25	toString . . . . .	2563
6.516.2.26	visit . . . . .	2563
6.516.3	Field Documentation . . . . .	2564
6.516.3.1	ackType . . . . .	2564
6.516.3.2	consumerId . . . . .	2564
6.516.3.3	destination . . . . .	2564
6.516.3.4	firstMessageId . . . . .	2564
6.516.3.5	ID_MESSAGEACK . . . . .	2564
6.516.3.6	lastMessageId . . . . .	2564
6.516.3.7	messageCount . . . . .	2564
6.516.3.8	transactionId . . . . .	2564
6.517	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class Reference	2565
6.517.1	Detailed Description . . . . .	2565
6.517.2	Constructor & Destructor Documentation . . . . .	2566
6.517.2.1	MessageAckMarshaller . . . . .	2566
6.517.2.2	~MessageAckMarshaller . . . . .	2566
6.517.3	Member Function Documentation . . . . .	2566
6.517.3.1	createObject . . . . .	2566
6.517.3.2	getDataStructureType . . . . .	2566
6.517.3.3	looseMarshal . . . . .	2566
6.517.3.4	looseUnmarshal . . . . .	2567
6.517.3.5	tightMarshal1 . . . . .	2567
6.517.3.6	tightMarshal2 . . . . .	2567
6.517.3.7	tightUnmarshal . . . . .	2568
6.518	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference	2569
6.518.1	Detailed Description . . . . .	2569
6.518.2	Constructor & Destructor Documentation . . . . .	2570
6.518.2.1	MessageAckMarshaller . . . . .	2570
6.518.2.2	~MessageAckMarshaller . . . . .	2570
6.518.3	Member Function Documentation . . . . .	2570
6.518.3.1	createObject . . . . .	2570

6.518.3.2	getDataStructureType . . . . .	2570
6.518.3.3	looseMarshal . . . . .	2570
6.518.3.4	looseUnmarshal . . . . .	2571
6.518.3.5	tightMarshal1 . . . . .	2571
6.518.3.6	tightMarshal2 . . . . .	2571
6.518.3.7	tightUnmarshal . . . . .	2572
6.519	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference	2573
6.519.1	Detailed Description . . . . .	2573
6.519.2	Constructor & Destructor Documentation . . . . .	2574
6.519.2.1	MessageAckMarshaller . . . . .	2574
6.519.2.2	~MessageAckMarshaller . . . . .	2574
6.519.3	Member Function Documentation . . . . .	2574
6.519.3.1	createObject . . . . .	2574
6.519.3.2	getDataStructureType . . . . .	2574
6.519.3.3	looseMarshal . . . . .	2574
6.519.3.4	looseUnmarshal . . . . .	2575
6.519.3.5	tightMarshal1 . . . . .	2575
6.519.3.6	tightMarshal2 . . . . .	2575
6.519.3.7	tightUnmarshal . . . . .	2576
6.520	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference	2577
6.520.1	Detailed Description . . . . .	2577
6.520.2	Constructor & Destructor Documentation . . . . .	2578
6.520.2.1	MessageAckMarshaller . . . . .	2578
6.520.2.2	~MessageAckMarshaller . . . . .	2578
6.520.3	Member Function Documentation . . . . .	2578
6.520.3.1	createObject . . . . .	2578
6.520.3.2	getDataStructureType . . . . .	2578
6.520.3.3	looseMarshal . . . . .	2578
6.520.3.4	looseUnmarshal . . . . .	2579
6.520.3.5	tightMarshal1 . . . . .	2579
6.520.3.6	tightMarshal2 . . . . .	2579
6.520.3.7	tightUnmarshal . . . . .	2580
6.521	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference	2581
6.521.1	Detailed Description . . . . .	2581
6.521.2	Constructor & Destructor Documentation . . . . .	2582
6.521.2.1	MessageAckMarshaller . . . . .	2582

6.521.2.2	~MessageAckMarshaller	2582
6.521.3	Member Function Documentation	2582
6.521.3.1	createObject	2582
6.521.3.2	getDataStructureType	2582
6.521.3.3	looseMarshal	2582
6.521.3.4	looseUnmarshal	2583
6.521.3.5	tightMarshal1	2583
6.521.3.6	tightMarshal2	2583
6.521.3.7	tightUnmarshal	2584
6.522	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference	2585
6.522.1	Detailed Description	2585
6.522.2	Constructor & Destructor Documentation	2586
6.522.2.1	MessageAckMarshaller	2586
6.522.2.2	~MessageAckMarshaller	2586
6.522.3	Member Function Documentation	2586
6.522.3.1	createObject	2586
6.522.3.2	getDataStructureType	2586
6.522.3.3	looseMarshal	2586
6.522.3.4	looseUnmarshal	2587
6.522.3.5	tightMarshal1	2587
6.522.3.6	tightMarshal2	2587
6.522.3.7	tightUnmarshal	2588
6.523	cms::MessageConsumer Class Reference	2589
6.523.1	Detailed Description	2589
6.523.2	Constructor & Destructor Documentation	2590
6.523.2.1	~MessageConsumer	2590
6.523.3	Member Function Documentation	2590
6.523.3.1	getMessageListener	2590
6.523.3.2	getMessageSelector	2590
6.523.3.3	receive	2590
6.523.3.4	receive	2591
6.523.3.5	receiveNoWait	2591
6.523.3.6	setMessageListener	2591
6.524	activemq::cmsutil::MessageCreator Class Reference	2593
6.524.1	Detailed Description	2593
6.524.2	Constructor & Destructor Documentation	2593

6.524.2.1	~MessageCreator	2593
6.524.3	Member Function Documentation	2593
6.524.3.1	createMessage	2593
6.525	activemq::commands::MessageDispatch Class Reference	2594
6.525.1	Constructor & Destructor Documentation	2595
6.525.1.1	MessageDispatch	2595
6.525.1.2	~MessageDispatch	2595
6.525.2	Member Function Documentation	2595
6.525.2.1	cloneDataStructure	2595
6.525.2.2	copyDataStructure	2595
6.525.2.3	equals	2595
6.525.2.4	getConsumerId	2596
6.525.2.5	getConsumerId	2596
6.525.2.6	getDataStructureType	2596
6.525.2.7	getDestination	2596
6.525.2.8	getDestination	2596
6.525.2.9	getMessage	2596
6.525.2.10	getMessage	2596
6.525.2.11	getRedeliveryCounter	2596
6.525.2.12	isMessageDispatch	2596
6.525.2.13	setConsumerId	2597
6.525.2.14	setDestination	2597
6.525.2.15	setMessage	2597
6.525.2.16	setRedeliveryCounter	2597
6.525.2.17	toString	2597
6.525.2.18	visit	2597
6.525.3	Field Documentation	2598
6.525.3.1	consumerId	2598
6.525.3.2	destination	2598
6.525.3.3	ID_MESSAGE_DISPATCH	2598
6.525.3.4	message	2598
6.525.3.5	redeliveryCounter	2598
6.526	activemq::core::MessageDispatchChannel Class Reference	2599
6.526.1	Constructor & Destructor Documentation	2600
6.526.1.1	MessageDispatchChannel	2600
6.526.1.2	~MessageDispatchChannel	2600

6.526.2 Member Function Documentation . . . . .	2600
6.526.2.1 clear . . . . .	2600
6.526.2.2 close . . . . .	2600
6.526.2.3 dequeue . . . . .	2601
6.526.2.4 dequeueNoWait . . . . .	2601
6.526.2.5 enqueue . . . . .	2601
6.526.2.6 enqueueFirst . . . . .	2601
6.526.2.7 isClosed . . . . .	2601
6.526.2.8 isEmpty . . . . .	2602
6.526.2.9 isRunning . . . . .	2602
6.526.2.10 lock . . . . .	2602
6.526.2.11 notify . . . . .	2602
6.526.2.12 notifyAll . . . . .	2602
6.526.2.13 peek . . . . .	2603
6.526.2.14 removeAll . . . . .	2603
6.526.2.15 size . . . . .	2603
6.526.2.16 start . . . . .	2603
6.526.2.17 stop . . . . .	2603
6.526.2.18 tryLock . . . . .	2603
6.526.2.19 unlock . . . . .	2604
6.526.2.20 wait . . . . .	2604
6.526.2.21 wait . . . . .	2604
6.526.2.22 wait . . . . .	2605
6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class	
Reference . . . . .	2606
6.527.1 Detailed Description . . . . .	2606
6.527.2 Constructor & Destructor Documentation . . . . .	2607
6.527.2.1 MessageDispatchMarshaller . . . . .	2607
6.527.2.2 ~MessageDispatchMarshaller . . . . .	2607
6.527.3 Member Function Documentation . . . . .	2607
6.527.3.1 createObject . . . . .	2607
6.527.3.2 getDataStructureType . . . . .	2607
6.527.3.3 looseMarshal . . . . .	2607
6.527.3.4 looseUnmarshal . . . . .	2608
6.527.3.5 tightMarshal1 . . . . .	2608
6.527.3.6 tightMarshal2 . . . . .	2608
6.527.3.7 tightUnmarshal . . . . .	2609

6.528	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	Class	
	Reference		2610
6.528.1	Detailed Description		2610
6.528.2	Constructor & Destructor Documentation		2611
	6.528.2.1 MessageDispatchMarshaller		2611
	6.528.2.2 ~MessageDispatchMarshaller		2611
6.528.3	Member Function Documentation		2611
	6.528.3.1 createObject		2611
	6.528.3.2 getDataStructureType		2611
	6.528.3.3 looseMarshal		2611
	6.528.3.4 looseUnmarshal		2612
	6.528.3.5 tightMarshal1		2612
	6.528.3.6 tightMarshal2		2612
	6.528.3.7 tightUnmarshal		2613
6.529	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	Class	
	Reference		2614
6.529.1	Detailed Description		2614
6.529.2	Constructor & Destructor Documentation		2615
	6.529.2.1 MessageDispatchMarshaller		2615
	6.529.2.2 ~MessageDispatchMarshaller		2615
6.529.3	Member Function Documentation		2615
	6.529.3.1 createObject		2615
	6.529.3.2 getDataStructureType		2615
	6.529.3.3 looseMarshal		2615
	6.529.3.4 looseUnmarshal		2616
	6.529.3.5 tightMarshal1		2616
	6.529.3.6 tightMarshal2		2616
	6.529.3.7 tightUnmarshal		2617
6.530	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	Class	
	Reference		2618
6.530.1	Detailed Description		2618
6.530.2	Constructor & Destructor Documentation		2619
	6.530.2.1 MessageDispatchMarshaller		2619
	6.530.2.2 ~MessageDispatchMarshaller		2619
6.530.3	Member Function Documentation		2619
	6.530.3.1 createObject		2619
	6.530.3.2 getDataStructureType		2619

6.530.3.3 looseMarshal . . . . .	2619
6.530.3.4 looseUnmarshal . . . . .	2620
6.530.3.5 tightMarshal1 . . . . .	2620
6.530.3.6 tightMarshal2 . . . . .	2620
6.530.3.7 tightUnmarshal . . . . .	2621
6.531activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class	
Reference . . . . .	2622
6.531.1 Detailed Description . . . . .	2622
6.531.2 Constructor & Destructor Documentation . . . . .	2623
6.531.2.1 MessageDispatchMarshaller . . . . .	2623
6.531.2.2 ~MessageDispatchMarshaller . . . . .	2623
6.531.3 Member Function Documentation . . . . .	2623
6.531.3.1 createObject . . . . .	2623
6.531.3.2 getDataStructureType . . . . .	2623
6.531.3.3 looseMarshal . . . . .	2623
6.531.3.4 looseUnmarshal . . . . .	2624
6.531.3.5 tightMarshal1 . . . . .	2624
6.531.3.6 tightMarshal2 . . . . .	2624
6.531.3.7 tightUnmarshal . . . . .	2625
6.532activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller Class	
Reference . . . . .	2626
6.532.1 Detailed Description . . . . .	2626
6.532.2 Constructor & Destructor Documentation . . . . .	2627
6.532.2.1 MessageDispatchMarshaller . . . . .	2627
6.532.2.2 ~MessageDispatchMarshaller . . . . .	2627
6.532.3 Member Function Documentation . . . . .	2627
6.532.3.1 createObject . . . . .	2627
6.532.3.2 getDataStructureType . . . . .	2627
6.532.3.3 looseMarshal . . . . .	2627
6.532.3.4 looseUnmarshal . . . . .	2628
6.532.3.5 tightMarshal1 . . . . .	2628
6.532.3.6 tightMarshal2 . . . . .	2628
6.532.3.7 tightUnmarshal . . . . .	2629
6.533activemq::commands::MessageDispatchNotification Class Reference . . . . .	2630
6.533.1 Constructor & Destructor Documentation . . . . .	2631
6.533.1.1 MessageDispatchNotification . . . . .	2631
6.533.1.2 ~MessageDispatchNotification . . . . .	2631

6.533.2 Member Function Documentation . . . . .	2631
6.533.2.1 cloneDataStructure . . . . .	2631
6.533.2.2 copyDataStructure . . . . .	2631
6.533.2.3 equals . . . . .	2631
6.533.2.4 getConsumerId . . . . .	2632
6.533.2.5 getConsumerId . . . . .	2632
6.533.2.6 getDataStructureType . . . . .	2632
6.533.2.7 getDeliverySequenceId . . . . .	2632
6.533.2.8 getDestination . . . . .	2632
6.533.2.9 getDestination . . . . .	2632
6.533.2.10 getMessageId . . . . .	2632
6.533.2.11 getMessageId . . . . .	2632
6.533.2.12 sMessageDispatchNotification . . . . .	2632
6.533.2.13 setConsumerId . . . . .	2633
6.533.2.14 setDeliverySequenceId . . . . .	2633
6.533.2.15 setDestination . . . . .	2633
6.533.2.16 setMessageId . . . . .	2633
6.533.2.17 toString . . . . .	2633
6.533.2.18 visit . . . . .	2633
6.533.3 Field Documentation . . . . .	2634
6.533.3.1 consumerId . . . . .	2634
6.533.3.2 deliverySequenceId . . . . .	2634
6.533.3.3 destination . . . . .	2634
6.533.3.4 ID_MESSAGE_DISPATCH_NOTIFICATION . . . . .	2634
6.533.3.5 messageId . . . . .	2634
6.534 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2635
6.534.1 Detailed Description . . . . .	2635
6.534.2 Constructor & Destructor Documentation . . . . .	2636
6.534.2.1 MessageDispatchNotificationMarshaller . . . . .	2636
6.534.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2636
6.534.3 Member Function Documentation . . . . .	2636
6.534.3.1 createObject . . . . .	2636
6.534.3.2 getDataStructureType . . . . .	2636
6.534.3.3 looseMarshal . . . . .	2636
6.534.3.4 looseUnmarshal . . . . .	2637
6.534.3.5 tightMarshal1 . . . . .	2637



6.534.3.6 tightMarshal2 . . . . .	2637
6.534.3.7 tightUnmarshal . . . . .	2638
6.535activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2639
6.535.1 Detailed Description . . . . .	2639
6.535.2 Constructor & Destructor Documentation . . . . .	2640
6.535.2.1 MessageDispatchNotificationMarshaller . . . . .	2640
6.535.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2640
6.535.3 Member Function Documentation . . . . .	2640
6.535.3.1 createObject . . . . .	2640
6.535.3.2 getDataStructureType . . . . .	2640
6.535.3.3 looseMarshal . . . . .	2640
6.535.3.4 looseUnmarshal . . . . .	2641
6.535.3.5 tightMarshal1 . . . . .	2641
6.535.3.6 tightMarshal2 . . . . .	2641
6.535.3.7 tightUnmarshal . . . . .	2642
6.536activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2643
6.536.1 Detailed Description . . . . .	2643
6.536.2 Constructor & Destructor Documentation . . . . .	2644
6.536.2.1 MessageDispatchNotificationMarshaller . . . . .	2644
6.536.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2644
6.536.3 Member Function Documentation . . . . .	2644
6.536.3.1 createObject . . . . .	2644
6.536.3.2 getDataStructureType . . . . .	2644
6.536.3.3 looseMarshal . . . . .	2644
6.536.3.4 looseUnmarshal . . . . .	2645
6.536.3.5 tightMarshal1 . . . . .	2645
6.536.3.6 tightMarshal2 . . . . .	2645
6.536.3.7 tightUnmarshal . . . . .	2646
6.537activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2647
6.537.1 Detailed Description . . . . .	2647
6.537.2 Constructor & Destructor Documentation . . . . .	2648
6.537.2.1 MessageDispatchNotificationMarshaller . . . . .	2648
6.537.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2648
6.537.3 Member Function Documentation . . . . .	2648

6.537.3.1	createObject . . . . .	2648
6.537.3.2	getDataStructureType . . . . .	2648
6.537.3.3	looseMarshal . . . . .	2648
6.537.3.4	looseUnmarshal . . . . .	2649
6.537.3.5	tightMarshal1 . . . . .	2649
6.537.3.6	tightMarshal2 . . . . .	2649
6.537.3.7	tightUnmarshal . . . . .	2650
6.538	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
	Class Reference . . . . .	2651
6.538.1	Detailed Description . . . . .	2651
6.538.2	Constructor & Destructor Documentation . . . . .	2652
	6.538.2.1 MessageDispatchNotificationMarshaller . . . . .	2652
	6.538.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2652
6.538.3	Member Function Documentation . . . . .	2652
	6.538.3.1 createObject . . . . .	2652
	6.538.3.2 getDataStructureType . . . . .	2652
	6.538.3.3 looseMarshal . . . . .	2652
	6.538.3.4 looseUnmarshal . . . . .	2653
	6.538.3.5 tightMarshal1 . . . . .	2653
	6.538.3.6 tightMarshal2 . . . . .	2653
	6.538.3.7 tightUnmarshal . . . . .	2654
6.539	activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	
	Class Reference . . . . .	2655
6.539.1	Detailed Description . . . . .	2655
6.539.2	Constructor & Destructor Documentation . . . . .	2656
	6.539.2.1 MessageDispatchNotificationMarshaller . . . . .	2656
	6.539.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2656
6.539.3	Member Function Documentation . . . . .	2656
	6.539.3.1 createObject . . . . .	2656
	6.539.3.2 getDataStructureType . . . . .	2656
	6.539.3.3 looseMarshal . . . . .	2656
	6.539.3.4 looseUnmarshal . . . . .	2657
	6.539.3.5 tightMarshal1 . . . . .	2657
	6.539.3.6 tightMarshal2 . . . . .	2657
	6.539.3.7 tightUnmarshal . . . . .	2658
6.540	cms::MessageEnumeration Class Reference . . . . .	2659
6.540.1	Detailed Description . . . . .	2659

6.540.2 Constructor & Destructor Documentation . . . . .	2659
6.540.2.1 ~MessageEnumeration . . . . .	2659
6.540.3 Member Function Documentation . . . . .	2659
6.540.3.1 hasMoreMessages . . . . .	2659
6.540.3.2 nextMessage . . . . .	2660
6.541cms::MessageEOFException Class Reference . . . . .	2661
6.541.1 Detailed Description . . . . .	2661
6.541.2 Constructor & Destructor Documentation . . . . .	2661
6.541.2.1 MessageEOFException . . . . .	2661
6.541.2.2 MessageEOFException . . . . .	2661
6.541.2.3 MessageEOFException . . . . .	2661
6.541.2.4 MessageEOFException . . . . .	2661
6.541.2.5 ~MessageEOFException . . . . .	2661
6.542cms::MessageFormatException Class Reference . . . . .	2662
6.542.1 Detailed Description . . . . .	2662
6.542.2 Constructor & Destructor Documentation . . . . .	2662
6.542.2.1 MessageFormatException . . . . .	2662
6.542.2.2 MessageFormatException . . . . .	2662
6.542.2.3 MessageFormatException . . . . .	2662
6.542.2.4 MessageFormatException . . . . .	2662
6.542.2.5 ~MessageFormatException . . . . .	2662
6.543activemq::commands::MessageId Class Reference . . . . .	2663
6.543.1 Member Typedef Documentation . . . . .	2664
6.543.1.1 COMPARATOR . . . . .	2664
6.543.2 Constructor & Destructor Documentation . . . . .	2664
6.543.2.1 MessageId . . . . .	2664
6.543.2.2 MessageId . . . . .	2664
6.543.2.3 MessageId . . . . .	2664
6.543.2.4 MessageId . . . . .	2664
6.543.2.5 MessageId . . . . .	2664
6.543.2.6 MessageId . . . . .	2664
6.543.2.7 ~MessageId . . . . .	2664
6.543.3 Member Function Documentation . . . . .	2664
6.543.3.1 cloneDataStructure . . . . .	2664
6.543.3.2 compareTo . . . . .	2665
6.543.3.3 copyDataStructure . . . . .	2665

6.543.3.4 equals . . . . .	2665
6.543.3.5 equals . . . . .	2665
6.543.3.6 getBrokerSequenceId . . . . .	2665
6.543.3.7 getDataStructureType . . . . .	2665
6.543.3.8 getProducerId . . . . .	2666
6.543.3.9 getProducerId . . . . .	2666
6.543.3.10getProducerSequenceId . . . . .	2666
6.543.3.11operator< . . . . .	2666
6.543.3.12operator= . . . . .	2666
6.543.3.13operator== . . . . .	2666
6.543.3.14setBrokerSequenceId . . . . .	2666
6.543.3.15setProducerId . . . . .	2666
6.543.3.16setProducerSequenceId . . . . .	2666
6.543.3.17set TextView . . . . .	2666
6.543.3.18set Value . . . . .	2666
6.543.3.19oString . . . . .	2666
6.543.4 Field Documentation . . . . .	2667
6.543.4.1 brokerSequenceId . . . . .	2667
6.543.4.2 ID_MESSAGEID . . . . .	2667
6.543.4.3 producerId . . . . .	2667
6.543.4.4 producerSequenceId . . . . .	2667
6.544activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference	2668
6.544.1 Detailed Description . . . . .	2668
6.544.2 Constructor & Destructor Documentation . . . . .	2669
6.544.2.1 MessageIdMarshaller . . . . .	2669
6.544.2.2 ~MessageIdMarshaller . . . . .	2669
6.544.3 Member Function Documentation . . . . .	2669
6.544.3.1 createObject . . . . .	2669
6.544.3.2 getDataStructureType . . . . .	2669
6.544.3.3 looseMarshal . . . . .	2669
6.544.3.4 looseUnmarshal . . . . .	2670
6.544.3.5 tightMarshal1 . . . . .	2670
6.544.3.6 tightMarshal2 . . . . .	2670
6.544.3.7 tightUnmarshal . . . . .	2671
6.545activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference	2672
6.545.1 Detailed Description . . . . .	2672

6.545.2 Constructor & Destructor Documentation . . . . .	2673
6.545.2.1 MessageIdMarshaller . . . . .	2673
6.545.2.2 ~MessageIdMarshaller . . . . .	2673
6.545.3 Member Function Documentation . . . . .	2673
6.545.3.1 createObject . . . . .	2673
6.545.3.2 getDataStructureType . . . . .	2673
6.545.3.3 looseMarshal . . . . .	2673
6.545.3.4 looseUnmarshal . . . . .	2674
6.545.3.5 tightMarshal1 . . . . .	2674
6.545.3.6 tightMarshal2 . . . . .	2674
6.545.3.7 tightUnmarshal . . . . .	2675
6.546activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference	2676
6.546.1 Detailed Description . . . . .	2676
6.546.2 Constructor & Destructor Documentation . . . . .	2677
6.546.2.1 MessageIdMarshaller . . . . .	2677
6.546.2.2 ~MessageIdMarshaller . . . . .	2677
6.546.3 Member Function Documentation . . . . .	2677
6.546.3.1 createObject . . . . .	2677
6.546.3.2 getDataStructureType . . . . .	2677
6.546.3.3 looseMarshal . . . . .	2677
6.546.3.4 looseUnmarshal . . . . .	2678
6.546.3.5 tightMarshal1 . . . . .	2678
6.546.3.6 tightMarshal2 . . . . .	2678
6.546.3.7 tightUnmarshal . . . . .	2679
6.547activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference	2680
6.547.1 Detailed Description . . . . .	2680
6.547.2 Constructor & Destructor Documentation . . . . .	2681
6.547.2.1 MessageIdMarshaller . . . . .	2681
6.547.2.2 ~MessageIdMarshaller . . . . .	2681
6.547.3 Member Function Documentation . . . . .	2681
6.547.3.1 createObject . . . . .	2681
6.547.3.2 getDataStructureType . . . . .	2681
6.547.3.3 looseMarshal . . . . .	2681
6.547.3.4 looseUnmarshal . . . . .	2682
6.547.3.5 tightMarshal1 . . . . .	2682
6.547.3.6 tightMarshal2 . . . . .	2682

6.547.3.7 tightUnmarshal . . . . .	2683
6.548activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference	2684
6.548.1 Detailed Description . . . . .	2684
6.548.2 Constructor & Destructor Documentation . . . . .	2685
6.548.2.1 MessageIdMarshaller . . . . .	2685
6.548.2.2 ~MessageIdMarshaller . . . . .	2685
6.548.3 Member Function Documentation . . . . .	2685
6.548.3.1 createObject . . . . .	2685
6.548.3.2 getDataStructureType . . . . .	2685
6.548.3.3 looseMarshal . . . . .	2685
6.548.3.4 looseUnmarshal . . . . .	2686
6.548.3.5 tightMarshal1 . . . . .	2686
6.548.3.6 tightMarshal2 . . . . .	2686
6.548.3.7 tightUnmarshal . . . . .	2687
6.549activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference	2688
6.549.1 Detailed Description . . . . .	2688
6.549.2 Constructor & Destructor Documentation . . . . .	2689
6.549.2.1 MessageIdMarshaller . . . . .	2689
6.549.2.2 ~MessageIdMarshaller . . . . .	2689
6.549.3 Member Function Documentation . . . . .	2689
6.549.3.1 createObject . . . . .	2689
6.549.3.2 getDataStructureType . . . . .	2689
6.549.3.3 looseMarshal . . . . .	2689
6.549.3.4 looseUnmarshal . . . . .	2690
6.549.3.5 tightMarshal1 . . . . .	2690
6.549.3.6 tightMarshal2 . . . . .	2690
6.549.3.7 tightUnmarshal . . . . .	2691
6.550cms::MessageListener Class Reference . . . . .	2692
6.550.1 Detailed Description . . . . .	2692
6.550.2 Constructor & Destructor Documentation . . . . .	2692
6.550.2.1 ~MessageListener . . . . .	2692
6.550.3 Member Function Documentation . . . . .	2692
6.550.3.1 onMessage . . . . .	2692
6.551activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference	2693
6.551.1 Detailed Description . . . . .	2693
6.551.2 Constructor & Destructor Documentation . . . . .	2694

6.551.2.1 MessageMarshaller . . . . .	2694
6.551.2.2 ~MessageMarshaller . . . . .	2694
6.551.3 Member Function Documentation . . . . .	2694
6.551.3.1 looseMarshal . . . . .	2694
6.551.3.2 looseUnmarshal . . . . .	2694
6.551.3.3 tightMarshal1 . . . . .	2695
6.551.3.4 tightMarshal2 . . . . .	2696
6.551.3.5 tightUnmarshal . . . . .	2696
6.552activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference . . . . .	2698
6.552.1 Detailed Description . . . . .	2698
6.552.2 Constructor & Destructor Documentation . . . . .	2699
6.552.2.1 MessageMarshaller . . . . .	2699
6.552.2.2 ~MessageMarshaller . . . . .	2699
6.552.3 Member Function Documentation . . . . .	2699
6.552.3.1 looseMarshal . . . . .	2699
6.552.3.2 looseUnmarshal . . . . .	2699
6.552.3.3 tightMarshal1 . . . . .	2700
6.552.3.4 tightMarshal2 . . . . .	2701
6.552.3.5 tightUnmarshal . . . . .	2701
6.553activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference . . . . .	2703
6.553.1 Detailed Description . . . . .	2703
6.553.2 Constructor & Destructor Documentation . . . . .	2704
6.553.2.1 MessageMarshaller . . . . .	2704
6.553.2.2 ~MessageMarshaller . . . . .	2704
6.553.3 Member Function Documentation . . . . .	2704
6.553.3.1 looseMarshal . . . . .	2704
6.553.3.2 looseUnmarshal . . . . .	2704
6.553.3.3 tightMarshal1 . . . . .	2705
6.553.3.4 tightMarshal2 . . . . .	2706
6.553.3.5 tightUnmarshal . . . . .	2706
6.554activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference . . . . .	2708
6.554.1 Detailed Description . . . . .	2708
6.554.2 Constructor & Destructor Documentation . . . . .	2709
6.554.2.1 MessageMarshaller . . . . .	2709
6.554.2.2 ~MessageMarshaller . . . . .	2709
6.554.3 Member Function Documentation . . . . .	2709

6.554.3.1 looseMarshal . . . . .	2709
6.554.3.2 looseUnmarshal . . . . .	2709
6.554.3.3 tightMarshal1 . . . . .	2710
6.554.3.4 tightMarshal2 . . . . .	2711
6.554.3.5 tightUnmarshal . . . . .	2711
6.555activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference . . . . .	2713
6.555.1 Detailed Description . . . . .	2713
6.555.2 Constructor & Destructor Documentation . . . . .	2714
6.555.2.1 MessageMarshaller . . . . .	2714
6.555.2.2 ~MessageMarshaller . . . . .	2714
6.555.3 Member Function Documentation . . . . .	2714
6.555.3.1 looseMarshal . . . . .	2714
6.555.3.2 looseUnmarshal . . . . .	2714
6.555.3.3 tightMarshal1 . . . . .	2715
6.555.3.4 tightMarshal2 . . . . .	2716
6.555.3.5 tightUnmarshal . . . . .	2716
6.556activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference . . . . .	2718
6.556.1 Detailed Description . . . . .	2718
6.556.2 Constructor & Destructor Documentation . . . . .	2719
6.556.2.1 MessageMarshaller . . . . .	2719
6.556.2.2 ~MessageMarshaller . . . . .	2719
6.556.3 Member Function Documentation . . . . .	2719
6.556.3.1 looseMarshal . . . . .	2719
6.556.3.2 looseUnmarshal . . . . .	2719
6.556.3.3 tightMarshal1 . . . . .	2720
6.556.3.4 tightMarshal2 . . . . .	2721
6.556.3.5 tightUnmarshal . . . . .	2721
6.557cms::MessageNotReadableException Class Reference . . . . .	2723
6.557.1 Detailed Description . . . . .	2723
6.557.2 Constructor & Destructor Documentation . . . . .	2723
6.557.2.1 MessageNotReadableException . . . . .	2723
6.557.2.2 MessageNotReadableException . . . . .	2723
6.557.2.3 MessageNotReadableException . . . . .	2723
6.557.2.4 MessageNotReadableException . . . . .	2723
6.557.2.5 ~MessageNotReadableException . . . . .	2723
6.558cms::MessageNotWriteableException Class Reference . . . . .	2724



6.558.1 Detailed Description . . . . .	2724
6.558.2 Constructor & Destructor Documentation . . . . .	2724
6.558.2.1 MessageNotWriteableException . . . . .	2724
6.558.2.2 MessageNotWriteableException . . . . .	2724
6.558.2.3 MessageNotWriteableException . . . . .	2724
6.558.2.4 MessageNotWriteableException . . . . .	2724
6.558.2.5 ~MessageNotWriteableException . . . . .	2724
6.559cms::MessageProducer Class Reference . . . . .	2725
6.559.1 Detailed Description . . . . .	2726
6.559.2 Constructor & Destructor Documentation . . . . .	2726
6.559.2.1 ~MessageProducer . . . . .	2726
6.559.3 Member Function Documentation . . . . .	2726
6.559.3.1 getDeliveryMode . . . . .	2726
6.559.3.2 getDisableMessageID . . . . .	2727
6.559.3.3 getDisableMessageTimeStamp . . . . .	2727
6.559.3.4 getPriority . . . . .	2727
6.559.3.5 getTimeToLive . . . . .	2728
6.559.3.6 send . . . . .	2728
6.559.3.7 send . . . . .	2728
6.559.3.8 send . . . . .	2729
6.559.3.9 send . . . . .	2729
6.559.3.10setDeliveryMode . . . . .	2730
6.559.3.11setDisableMessageID . . . . .	2730
6.559.3.12setDisableMessageTimeStamp . . . . .	2730
6.559.3.13setPriority . . . . .	2731
6.559.3.14setTimeToLive . . . . .	2731
6.560activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference	2732
6.560.1 Detailed Description . . . . .	2733
6.560.2 Constructor & Destructor Documentation . . . . .	2733
6.560.2.1 MessagePropertyInterceptor . . . . .	2733
6.560.2.2 ~MessagePropertyInterceptor . . . . .	2734
6.560.3 Member Function Documentation . . . . .	2734
6.560.3.1 getBooleanProperty . . . . .	2734
6.560.3.2 getByteProperty . . . . .	2734
6.560.3.3 getDoubleProperty . . . . .	2734
6.560.3.4 getFloatProperty . . . . .	2734

6.560.3.5	getIntProperty . . . . .	2735
6.560.3.6	getLongProperty . . . . .	2735
6.560.3.7	getShortProperty . . . . .	2735
6.560.3.8	getStringProperty . . . . .	2736
6.560.3.9	setBooleanProperty . . . . .	2736
6.560.3.10	setByteProperty . . . . .	2736
6.560.3.11	setDoubleProperty . . . . .	2736
6.560.3.12	setFloatProperty . . . . .	2737
6.560.3.13	setIntProperty . . . . .	2737
6.560.3.14	setLongProperty . . . . .	2737
6.560.3.15	setShortProperty . . . . .	2737
6.560.3.16	setStringProperty . . . . .	2737
6.561	activemq::commands::MessagePull Class Reference . . . . .	2739
6.561.1	Constructor & Destructor Documentation . . . . .	2740
6.561.1.1	MessagePull . . . . .	2740
6.561.1.2	~MessagePull . . . . .	2740
6.561.2	Member Function Documentation . . . . .	2740
6.561.2.1	cloneDataStructure . . . . .	2740
6.561.2.2	copyDataStructure . . . . .	2740
6.561.2.3	equals . . . . .	2740
6.561.2.4	getConsumerId . . . . .	2741
6.561.2.5	getConsumerId . . . . .	2741
6.561.2.6	getCorrelationId . . . . .	2741
6.561.2.7	getCorrelationId . . . . .	2741
6.561.2.8	getDataStructureType . . . . .	2741
6.561.2.9	getDestination . . . . .	2742
6.561.2.10	getDestination . . . . .	2742
6.561.2.11	getMessageId . . . . .	2742
6.561.2.12	getMessageId . . . . .	2742
6.561.2.13	getTimeout . . . . .	2742
6.561.2.14	setConsumerId . . . . .	2742
6.561.2.15	setCorrelationId . . . . .	2742
6.561.2.16	setDestination . . . . .	2742
6.561.2.17	setMessageId . . . . .	2742
6.561.2.18	setTimeout . . . . .	2742
6.561.2.19	toString . . . . .	2742

6.561.2.20	visit . . . . .	2742
6.561.3	Field Documentation . . . . .	2743
6.561.3.1	consumerId . . . . .	2743
6.561.3.2	correlationId . . . . .	2743
6.561.3.3	destination . . . . .	2743
6.561.3.4	ID_MESSAGEPULL . . . . .	2743
6.561.3.5	messageId . . . . .	2743
6.561.3.6	timeout . . . . .	2743
6.562	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference	2744
6.562.1	Detailed Description . . . . .	2744
6.562.2	Constructor & Destructor Documentation . . . . .	2745
6.562.2.1	MessagePullMarshaller . . . . .	2745
6.562.2.2	~MessagePullMarshaller . . . . .	2745
6.562.3	Member Function Documentation . . . . .	2745
6.562.3.1	createObject . . . . .	2745
6.562.3.2	getDataStructureType . . . . .	2745
6.562.3.3	looseMarshal . . . . .	2745
6.562.3.4	looseUnmarshal . . . . .	2746
6.562.3.5	tightMarshal1 . . . . .	2746
6.562.3.6	tightMarshal2 . . . . .	2746
6.562.3.7	tightUnmarshal . . . . .	2747
6.563	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference	2748
6.563.1	Detailed Description . . . . .	2748
6.563.2	Constructor & Destructor Documentation . . . . .	2749
6.563.2.1	MessagePullMarshaller . . . . .	2749
6.563.2.2	~MessagePullMarshaller . . . . .	2749
6.563.3	Member Function Documentation . . . . .	2749
6.563.3.1	createObject . . . . .	2749
6.563.3.2	getDataStructureType . . . . .	2749
6.563.3.3	looseMarshal . . . . .	2749
6.563.3.4	looseUnmarshal . . . . .	2750
6.563.3.5	tightMarshal1 . . . . .	2750
6.563.3.6	tightMarshal2 . . . . .	2750
6.563.3.7	tightUnmarshal . . . . .	2751
6.564	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference	2752
6.564.1	Detailed Description . . . . .	2752

6.564.2 Constructor & Destructor Documentation . . . . .	2753
6.564.2.1 MessagePullMarshaller . . . . .	2753
6.564.2.2 ~MessagePullMarshaller . . . . .	2753
6.564.3 Member Function Documentation . . . . .	2753
6.564.3.1 createObject . . . . .	2753
6.564.3.2 getDataStructureType . . . . .	2753
6.564.3.3 looseMarshal . . . . .	2753
6.564.3.4 looseUnmarshal . . . . .	2754
6.564.3.5 tightMarshal1 . . . . .	2754
6.564.3.6 tightMarshal2 . . . . .	2754
6.564.3.7 tightUnmarshal . . . . .	2755
6.565activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference	2756
6.565.1 Detailed Description . . . . .	2756
6.565.2 Constructor & Destructor Documentation . . . . .	2757
6.565.2.1 MessagePullMarshaller . . . . .	2757
6.565.2.2 ~MessagePullMarshaller . . . . .	2757
6.565.3 Member Function Documentation . . . . .	2757
6.565.3.1 createObject . . . . .	2757
6.565.3.2 getDataStructureType . . . . .	2757
6.565.3.3 looseMarshal . . . . .	2757
6.565.3.4 looseUnmarshal . . . . .	2758
6.565.3.5 tightMarshal1 . . . . .	2758
6.565.3.6 tightMarshal2 . . . . .	2758
6.565.3.7 tightUnmarshal . . . . .	2759
6.566activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference	2760
6.566.1 Detailed Description . . . . .	2760
6.566.2 Constructor & Destructor Documentation . . . . .	2761
6.566.2.1 MessagePullMarshaller . . . . .	2761
6.566.2.2 ~MessagePullMarshaller . . . . .	2761
6.566.3 Member Function Documentation . . . . .	2761
6.566.3.1 createObject . . . . .	2761
6.566.3.2 getDataStructureType . . . . .	2761
6.566.3.3 looseMarshal . . . . .	2761
6.566.3.4 looseUnmarshal . . . . .	2762
6.566.3.5 tightMarshal1 . . . . .	2762
6.566.3.6 tightMarshal2 . . . . .	2762

6.566.3.7 tightUnmarshal . . . . .	2763
6.567activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference	2764
6.567.1 Detailed Description . . . . .	2764
6.567.2 Constructor & Destructor Documentation . . . . .	2765
6.567.2.1 MessagePullMarshaller . . . . .	2765
6.567.2.2 ~MessagePullMarshaller . . . . .	2765
6.567.3 Member Function Documentation . . . . .	2765
6.567.3.1 createObject . . . . .	2765
6.567.3.2 getDataStructureType . . . . .	2765
6.567.3.3 looseMarshal . . . . .	2765
6.567.3.4 looseUnmarshal . . . . .	2766
6.567.3.5 tightMarshal1 . . . . .	2766
6.567.3.6 tightMarshal2 . . . . .	2766
6.567.3.7 tightUnmarshal . . . . .	2767
6.568activemq::transport::mock::MockTransport Class Reference . . . . .	2768
6.568.1 Detailed Description . . . . .	2770
6.568.2 Constructor & Destructor Documentation . . . . .	2770
6.568.2.1 MockTransport . . . . .	2770
6.568.2.2 ~MockTransport . . . . .	2770
6.568.3 Member Function Documentation . . . . .	2770
6.568.3.1 close . . . . .	2770
6.568.3.2 fireCommand . . . . .	2771
6.568.3.3 fireException . . . . .	2771
6.568.3.4 getInstance . . . . .	2771
6.568.3.5 getNumReceivedMessageBeforeFail . . . . .	2771
6.568.3.6 getNumReceivedMessages . . . . .	2771
6.568.3.7 getNumSentKeepAlives . . . . .	2771
6.568.3.8 getNumSentKeepAlivesBeforeFail . . . . .	2771
6.568.3.9 getNumSentMessageBeforeFail . . . . .	2771
6.568.3.10getNumSentMessages . . . . .	2771
6.568.3.11getRemoteAddress . . . . .	2771
6.568.3.12getTransportListener . . . . .	2772
6.568.3.13getWireFormat . . . . .	2772
6.568.3.14sClosed . . . . .	2772
6.568.3.15sConnected . . . . .	2772
6.568.3.16sFailOnClose . . . . .	2773

6.568.3.17sFailOnKeepAliveSends . . . . .	2773
6.568.3.18sFailOnReceiveMessage . . . . .	2773
6.568.3.19sFailOnSendMessage . . . . .	2773
6.568.3.20sFailOnStart . . . . .	2773
6.568.3.21sFailOnStop . . . . .	2773
6.568.3.22sFault Tolerant . . . . .	2773
6.568.3.23narrow . . . . .	2773
6.568.3.24neway . . . . .	2773
6.568.3.25reconnect . . . . .	2774
6.568.3.26request . . . . .	2774
6.568.3.27request . . . . .	2775
6.568.3.28set FailOnClose . . . . .	2776
6.568.3.29set FailOnKeepAliveSends . . . . .	2776
6.568.3.30set FailOnReceiveMessage . . . . .	2776
6.568.3.31set FailOnSendMessage . . . . .	2776
6.568.3.32set FailOnStart . . . . .	2776
6.568.3.33set FailOnStop . . . . .	2776
6.568.3.34set NumReceivedMessageBeforeFail . . . . .	2776
6.568.3.35set NumReceivedMessages . . . . .	2776
6.568.3.36set NumSentKeepAlives . . . . .	2776
6.568.3.37set NumSentKeepAlivesBeforeFail . . . . .	2776
6.568.3.38set NumSentMessageBeforeFail . . . . .	2776
6.568.3.39set NumSentMessages . . . . .	2776
6.568.3.40set OutgoingListener . . . . .	2776
6.568.3.41set ResponseBuilder . . . . .	2777
6.568.3.42set TransportListener . . . . .	2777
6.568.3.43set WireFormat . . . . .	2777
6.568.3.44start . . . . .	2777
6.568.3.45stop . . . . .	2777
6.569activemq::transport::mock::MockTransportFactory Class Reference . . . . .	2779
6.569.1 Detailed Description . . . . .	2779
6.569.2 Constructor & Destructor Documentation . . . . .	2780
6.569.2.1 ~MockTransportFactory . . . . .	2780
6.569.3 Member Function Documentation . . . . .	2780
6.569.3.1 create . . . . .	2780
6.569.3.2 createComposite . . . . .	2780

6.569.3.3 doCreateComposite . . . . .	2780
6.570decaf::util::concurrent::Mutex Class Reference . . . . .	2782
6.570.1 Detailed Description . . . . .	2783
6.570.2 Constructor & Destructor Documentation . . . . .	2783
6.570.2.1 Mutex . . . . .	2783
6.570.2.2 ~Mutex . . . . .	2783
6.570.3 Member Function Documentation . . . . .	2783
6.570.3.1 lock . . . . .	2783
6.570.3.2 notify . . . . .	2783
6.570.3.3 notifyAll . . . . .	2784
6.570.3.4 tryLock . . . . .	2784
6.570.3.5 unlock . . . . .	2784
6.570.3.6 wait . . . . .	2785
6.570.3.7 wait . . . . .	2785
6.570.3.8 wait . . . . .	2786
6.571decaf::util::concurrent::MutexHandle Class Reference . . . . .	2787
6.571.1 Constructor & Destructor Documentation . . . . .	2787
6.571.1.1 MutexHandle . . . . .	2787
6.571.1.2 ~MutexHandle . . . . .	2787
6.571.1.3 MutexHandle . . . . .	2787
6.571.1.4 ~MutexHandle . . . . .	2787
6.571.2 Field Documentation . . . . .	2787
6.571.2.1 lock_count . . . . .	2787
6.571.2.2 lock_owner . . . . .	2787
6.571.2.3 mutex . . . . .	2787
6.571.2.4 mutex . . . . .	2787
6.572decaf::internal::util::concurrent::MutexImpl Class Reference . . . . .	2788
6.572.1 Member Function Documentation . . . . .	2788
6.572.1.1 create . . . . .	2788
6.572.1.2 destroy . . . . .	2788
6.572.1.3 lock . . . . .	2789
6.572.1.4 trylock . . . . .	2789
6.572.1.5 unlock . . . . .	2789
6.573decaf::internal::net::Network Class Reference . . . . .	2790
6.573.1 Detailed Description . . . . .	2790
6.573.2 Constructor & Destructor Documentation . . . . .	2791

6.573.2.1 Network . . . . .	2791
6.573.2.2 ~Network . . . . .	2791
6.573.3 Member Function Documentation . . . . .	2791
6.573.3.1 addAsResource . . . . .	2791
6.573.3.2 addNetworkResource . . . . .	2791
6.573.3.3 getNetworkRuntime . . . . .	2791
6.573.3.4 getRuntimeLock . . . . .	2791
6.573.3.5 initializeNetworking . . . . .	2791
6.573.3.6 shutdownNetworking . . . . .	2792
6.574activemq::commands::NetworkBridgeFilter Class Reference . . . . .	2793
6.574.1 Constructor & Destructor Documentation . . . . .	2794
6.574.1.1 NetworkBridgeFilter . . . . .	2794
6.574.1.2 ~NetworkBridgeFilter . . . . .	2794
6.574.2 Member Function Documentation . . . . .	2794
6.574.2.1 cloneDataStructure . . . . .	2794
6.574.2.2 copyDataStructure . . . . .	2794
6.574.2.3 equals . . . . .	2794
6.574.2.4 getDataStructureType . . . . .	2794
6.574.2.5 getNetworkBrokerId . . . . .	2795
6.574.2.6 getNetworkBrokerId . . . . .	2795
6.574.2.7 getNetworkTTL . . . . .	2795
6.574.2.8 setNetworkBrokerId . . . . .	2795
6.574.2.9 setNetworkTTL . . . . .	2795
6.574.2.10toString . . . . .	2795
6.574.3 Field Documentation . . . . .	2795
6.574.3.1 ID_NETWORKBRIDGEFILTER . . . . .	2795
6.574.3.2 networkBrokerId . . . . .	2795
6.574.3.3 networkTTL . . . . .	2795
6.575activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference . . . . .	2796
6.575.1 Detailed Description . . . . .	2796
6.575.2 Constructor & Destructor Documentation . . . . .	2797
6.575.2.1 NetworkBridgeFilterMarshaller . . . . .	2797
6.575.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2797
6.575.3 Member Function Documentation . . . . .	2797
6.575.3.1 createObject . . . . .	2797
6.575.3.2 getDataStructureType . . . . .	2797



6.575.3.3 looseMarshal . . . . .	2797
6.575.3.4 looseUnmarshal . . . . .	2798
6.575.3.5 tightMarshal1 . . . . .	2798
6.575.3.6 tightMarshal2 . . . . .	2798
6.575.3.7 tightUnmarshal . . . . .	2799
6.576activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller Class	
Reference . . . . .	2800
6.576.1 Detailed Description . . . . .	2800
6.576.2 Constructor & Destructor Documentation . . . . .	2801
6.576.2.1 NetworkBridgeFilterMarshaller . . . . .	2801
6.576.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2801
6.576.3 Member Function Documentation . . . . .	2801
6.576.3.1 createObject . . . . .	2801
6.576.3.2 getDataStructureType . . . . .	2801
6.576.3.3 looseMarshal . . . . .	2801
6.576.3.4 looseUnmarshal . . . . .	2802
6.576.3.5 tightMarshal1 . . . . .	2802
6.576.3.6 tightMarshal2 . . . . .	2802
6.576.3.7 tightUnmarshal . . . . .	2803
6.577activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class	
Reference . . . . .	2804
6.577.1 Detailed Description . . . . .	2804
6.577.2 Constructor & Destructor Documentation . . . . .	2805
6.577.2.1 NetworkBridgeFilterMarshaller . . . . .	2805
6.577.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2805
6.577.3 Member Function Documentation . . . . .	2805
6.577.3.1 createObject . . . . .	2805
6.577.3.2 getDataStructureType . . . . .	2805
6.577.3.3 looseMarshal . . . . .	2805
6.577.3.4 looseUnmarshal . . . . .	2806
6.577.3.5 tightMarshal1 . . . . .	2806
6.577.3.6 tightMarshal2 . . . . .	2806
6.577.3.7 tightUnmarshal . . . . .	2807
6.578activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class	
Reference . . . . .	2808
6.578.1 Detailed Description . . . . .	2808
6.578.2 Constructor & Destructor Documentation . . . . .	2809

6.578.2.1 NetworkBridgeFilterMarshaller . . . . .	2809
6.578.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2809
6.578.3 Member Function Documentation . . . . .	2809
6.578.3.1 createObject . . . . .	2809
6.578.3.2 getDataStructureType . . . . .	2809
6.578.3.3 looseMarshal . . . . .	2809
6.578.3.4 looseUnmarshal . . . . .	2810
6.578.3.5 tightMarshal1 . . . . .	2810
6.578.3.6 tightMarshal2 . . . . .	2810
6.578.3.7 tightUnmarshal . . . . .	2811
6.579activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class	
Reference . . . . .	2812
6.579.1 Detailed Description . . . . .	2812
6.579.2 Constructor & Destructor Documentation . . . . .	2813
6.579.2.1 NetworkBridgeFilterMarshaller . . . . .	2813
6.579.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2813
6.579.3 Member Function Documentation . . . . .	2813
6.579.3.1 createObject . . . . .	2813
6.579.3.2 getDataStructureType . . . . .	2813
6.579.3.3 looseMarshal . . . . .	2813
6.579.3.4 looseUnmarshal . . . . .	2814
6.579.3.5 tightMarshal1 . . . . .	2814
6.579.3.6 tightMarshal2 . . . . .	2814
6.579.3.7 tightUnmarshal . . . . .	2815
6.580activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class	
Reference . . . . .	2816
6.580.1 Detailed Description . . . . .	2816
6.580.2 Constructor & Destructor Documentation . . . . .	2817
6.580.2.1 NetworkBridgeFilterMarshaller . . . . .	2817
6.580.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2817
6.580.3 Member Function Documentation . . . . .	2817
6.580.3.1 createObject . . . . .	2817
6.580.3.2 getDataStructureType . . . . .	2817
6.580.3.3 looseMarshal . . . . .	2817
6.580.3.4 looseUnmarshal . . . . .	2818
6.580.3.5 tightMarshal1 . . . . .	2818
6.580.3.6 tightMarshal2 . . . . .	2818

6.580.3.7 tightUnmarshal . . . . .	2819
6.581decaf::net::NoRouteToHostException Class Reference . . . . .	2820
6.581.1 Constructor & Destructor Documentation . . . . .	2820
6.581.1.1 NoRouteToHostException . . . . .	2820
6.581.1.2 NoRouteToHostException . . . . .	2820
6.581.1.3 NoRouteToHostException . . . . .	2821
6.581.1.4 NoRouteToHostException . . . . .	2821
6.581.1.5 NoRouteToHostException . . . . .	2821
6.581.1.6 NoRouteToHostException . . . . .	2821
6.581.1.7 ~NoRouteToHostException . . . . .	2822
6.581.2 Member Function Documentation . . . . .	2822
6.581.2.1 clone . . . . .	2822
6.582decaf::security::NoSuchAlgorithmException Class Reference . . . . .	2823
6.582.1 Constructor & Destructor Documentation . . . . .	2823
6.582.1.1 NoSuchAlgorithmException . . . . .	2823
6.582.1.2 NoSuchAlgorithmException . . . . .	2823
6.582.1.3 NoSuchAlgorithmException . . . . .	2824
6.582.1.4 NoSuchAlgorithmException . . . . .	2824
6.582.1.5 NoSuchAlgorithmException . . . . .	2824
6.582.1.6 NoSuchAlgorithmException . . . . .	2824
6.582.1.7 ~NoSuchAlgorithmException . . . . .	2825
6.582.2 Member Function Documentation . . . . .	2825
6.582.2.1 clone . . . . .	2825
6.583decaf::lang::exceptions::NoSuchElementException Class Reference . . . . .	2826
6.583.1 Constructor & Destructor Documentation . . . . .	2826
6.583.1.1 NoSuchElementException . . . . .	2826
6.583.1.2 NoSuchElementException . . . . .	2826
6.583.1.3 NoSuchElementException . . . . .	2827
6.583.1.4 NoSuchElementException . . . . .	2827
6.583.1.5 NoSuchElementException . . . . .	2827
6.583.1.6 NoSuchElementException . . . . .	2827
6.583.1.7 ~NoSuchElementException . . . . .	2828
6.583.2 Member Function Documentation . . . . .	2828
6.583.2.1 clone . . . . .	2828
6.584decaf::security::NoSuchProviderException Class Reference . . . . .	2829
6.584.1 Constructor & Destructor Documentation . . . . .	2829

6.584.1.1 NoSuchProviderException . . . . .	2829
6.584.1.2 NoSuchProviderException . . . . .	2829
6.584.1.3 NoSuchProviderException . . . . .	2830
6.584.1.4 NoSuchProviderException . . . . .	2830
6.584.1.5 NoSuchProviderException . . . . .	2830
6.584.1.6 NoSuchProviderException . . . . .	2830
6.584.1.7 ~NoSuchProviderException . . . . .	2831
6.584.2 Member Function Documentation . . . . .	2831
6.584.2.1 clone . . . . .	2831
6.585decaf::lang::exceptions::NullPointerException Class Reference . . . . .	2832
6.585.1 Constructor & Destructor Documentation . . . . .	2832
6.585.1.1 NullPointerException . . . . .	2832
6.585.1.2 NullPointerException . . . . .	2832
6.585.1.3 NullPointerException . . . . .	2833
6.585.1.4 NullPointerException . . . . .	2833
6.585.1.5 NullPointerException . . . . .	2833
6.585.1.6 NullPointerException . . . . .	2833
6.585.1.7 ~NullPointerException . . . . .	2834
6.585.2 Member Function Documentation . . . . .	2834
6.585.2.1 clone . . . . .	2834
6.586decaf::lang::Number Class Reference . . . . .	2835
6.586.1 Detailed Description . . . . .	2835
6.586.2 Constructor & Destructor Documentation . . . . .	2835
6.586.2.1 ~Number . . . . .	2835
6.586.3 Member Function Documentation . . . . .	2835
6.586.3.1 byteValue . . . . .	2835
6.586.3.2 doubleValue . . . . .	2836
6.586.3.3 floatValue . . . . .	2836
6.586.3.4 intValue . . . . .	2836
6.586.3.5 longValue . . . . .	2836
6.586.3.6 shortValue . . . . .	2837
6.587decaf::lang::exceptions::NumberFormatException Class Reference . . . . .	2838
6.587.1 Constructor & Destructor Documentation . . . . .	2838
6.587.1.1 NumberFormatException . . . . .	2838
6.587.1.2 NumberFormatException . . . . .	2838
6.587.1.3 NumberFormatException . . . . .	2839

6.587.1.4 NumberFormatException . . . . .	2839
6.587.1.5 NumberFormatException . . . . .	2839
6.587.1.6 NumberFormatException . . . . .	2839
6.587.1.7 ~NumberFormatException . . . . .	2840
6.587.2 Member Function Documentation . . . . .	2840
6.587.2.1 clone . . . . .	2840
6.588cms::ObjectMessage Class Reference . . . . .	2841
6.588.1 Detailed Description . . . . .	2841
6.588.2 Constructor & Destructor Documentation . . . . .	2841
6.588.2.1 ~ObjectMessage . . . . .	2841
6.589decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference . . . . .	2842
6.589.1 Detailed Description . . . . .	2843
6.589.2 Constructor & Destructor Documentation . . . . .	2843
6.589.2.1 OpenSSLContextSpi . . . . .	2843
6.589.2.2 ~OpenSSLContextSpi . . . . .	2843
6.589.3 Member Function Documentation . . . . .	2843
6.589.3.1 providerGetServerSocketFactory . . . . .	2843
6.589.3.2 providerGetSocketFactory . . . . .	2843
6.589.3.3 providerInit . . . . .	2844
6.589.4 Friends And Related Function Documentation . . . . .	2844
6.589.4.1 OpenSSLSocket . . . . .	2844
6.589.4.2 OpenSSLSocketFactory . . . . .	2844
6.590decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference . . . . .	2845
6.590.1 Detailed Description . . . . .	2845
6.590.2 Constructor & Destructor Documentation . . . . .	2845
6.590.2.1 ~OpenSSLParameters . . . . .	2845
6.590.3 Member Function Documentation . . . . .	2845
6.590.3.1 clone . . . . .	2845
6.590.3.2 getEnabledCipherSuites . . . . .	2846
6.590.3.3 getEnabledProtocols . . . . .	2846
6.590.3.4 getNeedClientAuth . . . . .	2846
6.590.3.5 getSupportedCipherSuites . . . . .	2846
6.590.3.6 getSupportedProtocols . . . . .	2846
6.590.3.7 getUseClientMode . . . . .	2846
6.590.3.8 getWantClientAuth . . . . .	2846
6.590.3.9 setEnabledCipherSuites . . . . .	2846

6.590.3.10	setEnabledProtocols . . . . .	2846
6.590.3.11	setNeedClientAuth . . . . .	2846
6.590.3.12	setUseClientMode . . . . .	2846
6.590.3.13	setWantClientAuth . . . . .	2846
6.591	decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference . . . . .	2847
6.591.1	Detailed Description . . . . .	2848
6.591.2	Constructor & Destructor Documentation . . . . .	2849
6.591.2.1	OpenSSLServerSocket . . . . .	2849
6.591.2.2	~OpenSSLServerSocket . . . . .	2849
6.591.3	Member Function Documentation . . . . .	2849
6.591.3.1	accept . . . . .	2849
6.591.3.2	setEnabledCipherSuites . . . . .	2849
6.591.3.3	setEnabledProtocols . . . . .	2850
6.591.3.4	getNeedClientAuth . . . . .	2850
6.591.3.5	getSupportedCipherSuites . . . . .	2850
6.591.3.6	getSupportedProtocols . . . . .	2850
6.591.3.7	getWantClientAuth . . . . .	2851
6.591.3.8	setEnabledCipherSuites . . . . .	2851
6.591.3.9	setEnabledProtocols . . . . .	2851
6.591.3.10	setNeedClientAuth . . . . .	2851
6.591.3.11	setWantClientAuth . . . . .	2852
6.592	decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference . . . . .	2853
6.592.1	Detailed Description . . . . .	2854
6.592.2	Constructor & Destructor Documentation . . . . .	2855
6.592.2.1	OpenSSLServerSocketFactory . . . . .	2855
6.592.2.2	~OpenSSLServerSocketFactory . . . . .	2855
6.592.3	Member Function Documentation . . . . .	2855
6.592.3.1	createServerSocket . . . . .	2855
6.592.3.2	createServerSocket . . . . .	2855
6.592.3.3	createServerSocket . . . . .	2856
6.592.3.4	createServerSocket . . . . .	2856
6.592.3.5	getDefaultCipherSuites . . . . .	2856
6.592.3.6	getSupportedCipherSuites . . . . .	2857
6.593	decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference . . . . .	2858
6.593.1	Detailed Description . . . . .	2862
6.593.2	Constructor & Destructor Documentation . . . . .	2862

6.593.2.1	OpenSSLSocket . . . . .	2862
6.593.2.2	OpenSSLSocket . . . . .	2862
6.593.2.3	OpenSSLSocket . . . . .	2862
6.593.2.4	OpenSSLSocket . . . . .	2862
6.593.2.5	OpenSSLSocket . . . . .	2862
6.593.2.6	~OpenSSLSocket . . . . .	2862
6.593.3	Member Function Documentation . . . . .	2862
6.593.3.1	available . . . . .	2862
6.593.3.2	close . . . . .	2863
6.593.3.3	connect . . . . .	2863
6.593.3.4	getEnabledCipherSuites . . . . .	2863
6.593.3.5	getEnabledProtocols . . . . .	2864
6.593.3.6	getInputStream . . . . .	2864
6.593.3.7	getNeedClientAuth . . . . .	2864
6.593.3.8	getOutputStream . . . . .	2865
6.593.3.9	getSupportedCipherSuites . . . . .	2865
6.593.3.10	getSupportedProtocols . . . . .	2865
6.593.3.11	getUseClientMode . . . . .	2865
6.593.3.12	getWantClientAuth . . . . .	2866
6.593.3.13	read . . . . .	2866
6.593.3.14	endUrgentData . . . . .	2866
6.593.3.15	setEnabledCipherSuites . . . . .	2867
6.593.3.16	setEnabledProtocols . . . . .	2867
6.593.3.17	setNeedClientAuth . . . . .	2867
6.593.3.18	setOOBInline . . . . .	2868
6.593.3.19	setUseClientMode . . . . .	2868
6.593.3.20	setWantClientAuth . . . . .	2868
6.593.3.21	shutdownInput . . . . .	2869
6.593.3.22	shutdownOutput . . . . .	2869
6.593.3.23	startHandshake . . . . .	2869
6.593.3.24	write . . . . .	2869
6.594	decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference . . . . .	2871
6.594.1	Detailed Description . . . . .	2872
6.594.2	Constructor & Destructor Documentation . . . . .	2872
6.594.2.1	OpenSSLSocketException . . . . .	2872
6.594.2.2	OpenSSLSocketException . . . . .	2872

6.594.2.3 OpenSSLSocketException . . . . .	2872
6.594.2.4 OpenSSLSocketException . . . . .	2872
6.594.2.5 OpenSSLSocketException . . . . .	2873
6.594.2.6 OpenSSLSocketException . . . . .	2873
6.594.2.7 OpenSSLSocketException . . . . .	2873
6.594.2.8 ~OpenSSLSocketException . . . . .	2873
6.594.3 Member Function Documentation . . . . .	2873
6.594.3.1 clone . . . . .	2873
6.594.3.2 getErrorString . . . . .	2874
6.595decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference . . . . .	2875
6.595.1 Detailed Description . . . . .	2877
6.595.2 Constructor & Destructor Documentation . . . . .	2877
6.595.2.1 OpenSSLSocketFactory . . . . .	2877
6.595.2.2 ~OpenSSLSocketFactory . . . . .	2877
6.595.3 Member Function Documentation . . . . .	2877
6.595.3.1 createSocket . . . . .	2877
6.595.3.2 createSocket . . . . .	2878
6.595.3.3 createSocket . . . . .	2878
6.595.3.4 createSocket . . . . .	2879
6.595.3.5 createSocket . . . . .	2879
6.595.3.6 createSocket . . . . .	2880
6.595.3.7 getDefaultCipherSuites . . . . .	2880
6.595.3.8 getSupportedCipherSuites . . . . .	2880
6.596decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference . . . . .	2882
6.596.1 Detailed Description . . . . .	2882
6.596.2 Constructor & Destructor Documentation . . . . .	2883
6.596.2.1 OpenSSLSocketInputStream . . . . .	2883
6.596.2.2 ~OpenSSLSocketInputStream . . . . .	2883
6.596.3 Member Function Documentation . . . . .	2883
6.596.3.1 available . . . . .	2883
6.596.3.2 close . . . . .	2883
6.596.3.3 doReadArrayBounded . . . . .	2883
6.596.3.4 doReadByte . . . . .	2884
6.596.3.5 skip . . . . .	2884
6.597decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference . . . . .	2885
6.597.1 Detailed Description . . . . .	2885



6.597.2 Constructor & Destructor Documentation . . . . .	2886
6.597.2.1 OpenSSLSocketOutputStream . . . . .	2886
6.597.2.2 ~OpenSSLSocketOutputStream . . . . .	2886
6.597.3 Member Function Documentation . . . . .	2886
6.597.3.1 close . . . . .	2886
6.597.3.2 doWriteArrayBounded . . . . .	2886
6.597.3.3 doWriteByte . . . . .	2886
6.598activemq::wireformat::openwire::OpenWireFormat Class Reference . . . . .	2887
6.598.1 Constructor & Destructor Documentation . . . . .	2890
6.598.1.1 OpenWireFormat . . . . .	2890
6.598.1.2 ~OpenWireFormat . . . . .	2890
6.598.2 Member Function Documentation . . . . .	2890
6.598.2.1 addMarshaller . . . . .	2890
6.598.2.2 createNegotiator . . . . .	2890
6.598.2.3 destroyMarshalers . . . . .	2890
6.598.2.4 doUnmarshal . . . . .	2891
6.598.2.5 getCacheSize . . . . .	2891
6.598.2.6 getMaxInactivityDuration . . . . .	2891
6.598.2.7 getMaxInactivityDurationInitialDelay . . . . .	2891
6.598.2.8 getPreferredWireFormatInfo . . . . .	2892
6.598.2.9 getVersion . . . . .	2892
6.598.2.10hasNegotiator . . . . .	2892
6.598.2.11inReceive . . . . .	2892
6.598.2.12sCacheEnabled . . . . .	2892
6.598.2.13sSizePrefixDisabled . . . . .	2893
6.598.2.14sStackTraceEnabled . . . . .	2893
6.598.2.15sTcpNoDelayEnabled . . . . .	2893
6.598.2.16sTightEncodingEnabled . . . . .	2893
6.598.2.17ooseMarshalNestedObject . . . . .	2893
6.598.2.18ooseUnmarshalNestedObject . . . . .	2894
6.598.2.19marshal . . . . .	2894
6.598.2.20renegotiateWireFormat . . . . .	2894
6.598.2.21setCacheEnabled . . . . .	2895
6.598.2.22setCacheSize . . . . .	2895
6.598.2.23setMaxInactivityDuration . . . . .	2895
6.598.2.24setMaxInactivityDurationInitialDelay . . . . .	2895

6.598.2.25	setPreferedWireFormatInfo . . . . .	2895
6.598.2.26	setSizePrefixDisabled . . . . .	2896
6.598.2.27	setStackTraceEnabled . . . . .	2896
6.598.2.28	setTcpNoDelayEnabled . . . . .	2896
6.598.2.29	setTightEncodingEnabled . . . . .	2896
6.598.2.30	setVersion . . . . .	2896
6.598.2.31	tightMarshalNestedObject1 . . . . .	2897
6.598.2.32	tightMarshalNestedObject2 . . . . .	2897
6.598.2.33	tightUnmarshalNestedObject . . . . .	2897
6.598.2.34	unmarshal . . . . .	2898
6.598.3	Field Documentation . . . . .	2898
6.598.3.1	DEFAULT_VERSION . . . . .	2898
6.598.3.2	NULL_TYPE . . . . .	2898
6.599	activemq::wireformat::openwire::OpenWireFormatFactory Class Reference . . . . .	2899
6.599.1	Constructor & Destructor Documentation . . . . .	2899
6.599.1.1	OpenWireFormatFactory . . . . .	2899
6.599.1.2	~OpenWireFormatFactory . . . . .	2899
6.599.2	Member Function Documentation . . . . .	2899
6.599.2.1	createWireFormat . . . . .	2899
6.600	activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference . . . . .	2901
6.600.1	Constructor & Destructor Documentation . . . . .	2901
6.600.1.1	OpenWireFormatNegotiator . . . . .	2901
6.600.1.2	~OpenWireFormatNegotiator . . . . .	2902
6.600.2	Member Function Documentation . . . . .	2902
6.600.2.1	close . . . . .	2902
6.600.2.2	onCommand . . . . .	2902
6.600.2.3	oneway . . . . .	2902
6.600.2.4	onTransportException . . . . .	2903
6.600.2.5	request . . . . .	2903
6.600.2.6	request . . . . .	2903
6.600.2.7	start . . . . .	2904
6.601	activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference . . . . .	2905
6.601.1	Constructor & Destructor Documentation . . . . .	2905
6.601.1.1	OpenWireResponseBuilder . . . . .	2905
6.601.1.2	~OpenWireResponseBuilder . . . . .	2905
6.601.2	Member Function Documentation . . . . .	2905

6.601.2.1 buildIncomingCommands . . . . .	2905
6.601.2.2 buildResponse . . . . .	2906
6.602decaf::io::OutputStream Class Reference . . . . .	2907
6.602.1 Detailed Description . . . . .	2908
6.602.2 Constructor & Destructor Documentation . . . . .	2909
6.602.2.1 OutputStream . . . . .	2909
6.602.2.2 ~OutputStream . . . . .	2909
6.602.3 Member Function Documentation . . . . .	2909
6.602.3.1 close . . . . .	2909
6.602.3.2 doWriteArray . . . . .	2909
6.602.3.3 doWriteArrayBounded . . . . .	2909
6.602.3.4 doWriteByte . . . . .	2910
6.602.3.5 flush . . . . .	2910
6.602.3.6 lock . . . . .	2910
6.602.3.7 notify . . . . .	2910
6.602.3.8 notifyAll . . . . .	2911
6.602.3.9 toString . . . . .	2911
6.602.3.10tryLock . . . . .	2911
6.602.3.11unlock . . . . .	2911
6.602.3.12wait . . . . .	2912
6.602.3.13wait . . . . .	2912
6.602.3.14wait . . . . .	2913
6.602.3.15write . . . . .	2913
6.602.3.16write . . . . .	2913
6.602.3.17write . . . . .	2914
6.603decaf::io::OutputStreamWriter Class Reference . . . . .	2915
6.603.1 Detailed Description . . . . .	2915
6.603.2 Constructor & Destructor Documentation . . . . .	2915
6.603.2.1 OutputStreamWriter . . . . .	2915
6.603.2.2 ~OutputStreamWriter . . . . .	2916
6.603.3 Member Function Documentation . . . . .	2916
6.603.3.1 checkClosed . . . . .	2916
6.603.3.2 close . . . . .	2916
6.603.3.3 doWriteArrayBounded . . . . .	2916
6.603.3.4 flush . . . . .	2916
6.604activemq::commands::PartialCommand Class Reference . . . . .	2918

6.604.1 Constructor & Destructor Documentation . . . . .	2919
6.604.1.1 PartialCommand . . . . .	2919
6.604.1.2 ~PartialCommand . . . . .	2919
6.604.2 Member Function Documentation . . . . .	2919
6.604.2.1 cloneDataStructure . . . . .	2919
6.604.2.2 copyDataStructure . . . . .	2919
6.604.2.3 equals . . . . .	2919
6.604.2.4 getCommandId . . . . .	2920
6.604.2.5 getData . . . . .	2920
6.604.2.6 getData . . . . .	2920
6.604.2.7 getDataStructureType . . . . .	2920
6.604.2.8 setCommandId . . . . .	2920
6.604.2.9 setData . . . . .	2920
6.604.2.10 toString . . . . .	2920
6.604.3 Field Documentation . . . . .	2921
6.604.3.1 commandId . . . . .	2921
6.604.3.2 data . . . . .	2921
6.604.3.3 ID_PARTIALCOMMAND . . . . .	2921
6.605activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller      Class	
Reference . . . . .	2922
6.605.1 Detailed Description . . . . .	2922
6.605.2 Constructor & Destructor Documentation . . . . .	2923
6.605.2.1 PartialCommandMarshaller . . . . .	2923
6.605.2.2 ~PartialCommandMarshaller . . . . .	2923
6.605.3 Member Function Documentation . . . . .	2923
6.605.3.1 createObject . . . . .	2923
6.605.3.2 getDataStructureType . . . . .	2923
6.605.3.3 looseMarshal . . . . .	2923
6.605.3.4 looseUnmarshal . . . . .	2924
6.605.3.5 tightMarshal1 . . . . .	2924
6.605.3.6 tightMarshal2 . . . . .	2925
6.605.3.7 tightUnmarshal . . . . .	2925
6.606activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller      Class	
Reference . . . . .	2926
6.606.1 Detailed Description . . . . .	2926
6.606.2 Constructor & Destructor Documentation . . . . .	2927
6.606.2.1 PartialCommandMarshaller . . . . .	2927

6.606.2.2	~PartialCommandMarshaller . . . . .	2927
6.606.3	Member Function Documentation . . . . .	2927
6.606.3.1	createObject . . . . .	2927
6.606.3.2	getDataStructureType . . . . .	2927
6.606.3.3	looseMarshal . . . . .	2927
6.606.3.4	looseUnmarshal . . . . .	2928
6.606.3.5	tightMarshal1 . . . . .	2928
6.606.3.6	tightMarshal2 . . . . .	2929
6.606.3.7	tightUnmarshal . . . . .	2929
6.607	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller Class	
	Reference . . . . .	2930
6.607.1	Detailed Description . . . . .	2930
6.607.2	Constructor & Destructor Documentation . . . . .	2931
6.607.2.1	PartialCommandMarshaller . . . . .	2931
6.607.2.2	~PartialCommandMarshaller . . . . .	2931
6.607.3	Member Function Documentation . . . . .	2931
6.607.3.1	createObject . . . . .	2931
6.607.3.2	getDataStructureType . . . . .	2931
6.607.3.3	looseMarshal . . . . .	2931
6.607.3.4	looseUnmarshal . . . . .	2932
6.607.3.5	tightMarshal1 . . . . .	2932
6.607.3.6	tightMarshal2 . . . . .	2933
6.607.3.7	tightUnmarshal . . . . .	2933
6.608	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class	
	Reference . . . . .	2934
6.608.1	Detailed Description . . . . .	2934
6.608.2	Constructor & Destructor Documentation . . . . .	2935
6.608.2.1	PartialCommandMarshaller . . . . .	2935
6.608.2.2	~PartialCommandMarshaller . . . . .	2935
6.608.3	Member Function Documentation . . . . .	2935
6.608.3.1	createObject . . . . .	2935
6.608.3.2	getDataStructureType . . . . .	2935
6.608.3.3	looseMarshal . . . . .	2935
6.608.3.4	looseUnmarshal . . . . .	2936
6.608.3.5	tightMarshal1 . . . . .	2936
6.608.3.6	tightMarshal2 . . . . .	2937
6.608.3.7	tightUnmarshal . . . . .	2937

6.609	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	Class	
	Reference		2938
6.609.1	Detailed Description		2938
6.609.2	Constructor & Destructor Documentation		2939
	6.609.2.1 PartialCommandMarshaller		2939
	6.609.2.2 ~PartialCommandMarshaller		2939
6.609.3	Member Function Documentation		2939
	6.609.3.1 createObject		2939
	6.609.3.2 getDataStructureType		2939
	6.609.3.3 looseMarshal		2939
	6.609.3.4 looseUnmarshal		2940
	6.609.3.5 tightMarshal1		2940
	6.609.3.6 tightMarshal2		2941
	6.609.3.7 tightUnmarshal		2941
6.610	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	Class	
	Reference		2942
6.610.1	Detailed Description		2942
6.610.2	Constructor & Destructor Documentation		2943
	6.610.2.1 PartialCommandMarshaller		2943
	6.610.2.2 ~PartialCommandMarshaller		2943
6.610.3	Member Function Documentation		2943
	6.610.3.1 createObject		2943
	6.610.3.2 getDataStructureType		2943
	6.610.3.3 looseMarshal		2943
	6.610.3.4 looseUnmarshal		2944
	6.610.3.5 tightMarshal1		2944
	6.610.3.6 tightMarshal2		2945
	6.610.3.7 tightUnmarshal		2945
6.611	decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference		2946
	6.611.1 Detailed Description		2947
	6.611.2 Member Typedef Documentation		2948
	6.611.2.1 CounterType		2948
	6.611.2.2 PointerType		2948
	6.611.2.3 ReferenceType		2948
	6.611.3 Constructor & Destructor Documentation		2948
	6.611.3.1 Pointer		2948
	6.611.3.2 Pointer		2948

6.611.3.3 Pointer . . . . .	2948
6.611.3.4 Pointer . . . . .	2949
6.611.3.5 Pointer . . . . .	2949
6.611.3.6 Pointer . . . . .	2949
6.611.3.7 ~Pointer . . . . .	2950
6.611.4 Member Function Documentation . . . . .	2950
6.611.4.1 dynamicCast . . . . .	2950
6.611.4.2 get . . . . .	2950
6.611.4.3 operator! . . . . .	2950
6.611.4.4 operator!= . . . . .	2950
6.611.4.5 operator* . . . . .	2950
6.611.4.6 operator* . . . . .	2950
6.611.4.7 operator-> . . . . .	2951
6.611.4.8 operator-> . . . . .	2951
6.611.4.9 operator= . . . . .	2951
6.611.4.10operator= . . . . .	2951
6.611.4.11operator== . . . . .	2951
6.611.4.12release . . . . .	2951
6.611.4.13reset . . . . .	2952
6.611.4.14staticCast . . . . .	2952
6.611.4.15swap . . . . .	2952
6.611.5 Friends And Related Function Documentation . . . . .	2953
6.611.5.1 operator!= . . . . .	2953
6.611.5.2 operator!= . . . . .	2953
6.611.5.3 operator== . . . . .	2953
6.611.5.4 operator== . . . . .	2953
6.612decaf::lang::PointerComparator< T, R > Class Template Reference . . . . .	2954
6.612.1 Detailed Description . . . . .	2954
6.612.2 Member Function Documentation . . . . .	2954
6.612.2.1 compare . . . . .	2954
6.612.2.2 operator() . . . . .	2954
6.613activemq::cmsutil::PooledSession Class Reference . . . . .	2955
6.613.1 Detailed Description . . . . .	2957
6.613.2 Constructor & Destructor Documentation . . . . .	2957
6.613.2.1 PooledSession . . . . .	2957
6.613.2.2 PooledSession . . . . .	2957

6.613.2.3 ~PooledSession . . . . .	2957
6.613.3 Member Function Documentation . . . . .	2957
6.613.3.1 close . . . . .	2957
6.613.3.2 commit . . . . .	2958
6.613.3.3 createBrowser . . . . .	2958
6.613.3.4 createBrowser . . . . .	2958
6.613.3.5 createBytesMessage . . . . .	2959
6.613.3.6 createBytesMessage . . . . .	2959
6.613.3.7 createCachedConsumer . . . . .	2959
6.613.3.8 createCachedProducer . . . . .	2960
6.613.3.9 createConsumer . . . . .	2960
6.613.3.10 createConsumer . . . . .	2961
6.613.3.11 createConsumer . . . . .	2961
6.613.3.12 createDurableConsumer . . . . .	2961
6.613.3.13 createMapMessage . . . . .	2962
6.613.3.14 createMessage . . . . .	2962
6.613.3.15 createProducer . . . . .	2962
6.613.3.16 createQueue . . . . .	2963
6.613.3.17 createStreamMessage . . . . .	2963
6.613.3.18 createTemporaryQueue . . . . .	2963
6.613.3.19 createTemporaryTopic . . . . .	2964
6.613.3.20 createTextMessage . . . . .	2964
6.613.3.21 createTextMessage . . . . .	2964
6.613.3.22 createTopic . . . . .	2964
6.613.3.23 getAcknowledgeMode . . . . .	2965
6.613.3.24 getSession . . . . .	2965
6.613.3.25 getSession . . . . .	2965
6.613.3.26 isTransacted . . . . .	2965
6.613.3.27 operator= . . . . .	2966
6.613.3.28 recover . . . . .	2966
6.613.3.29 rollback . . . . .	2966
6.613.3.30 unsubscribe . . . . .	2966
6.614 decaf::util::concurrent::PooledThread Class Reference . . . . .	2968
6.614.1 Constructor & Destructor Documentation . . . . .	2968
6.614.1.1 PooledThread . . . . .	2968
6.614.1.2 ~PooledThread . . . . .	2969



6.614.2 Member Function Documentation . . . . .	2969
6.614.2.1 getPooledThreadListener . . . . .	2969
6.614.2.2 isBusy . . . . .	2969
6.614.2.3 run . . . . .	2969
6.614.2.4 setPooledThreadListener . . . . .	2969
6.614.2.5 stop . . . . .	2969
6.615decaf::util::concurrent::PooledThreadListener Class Reference . . . . .	2971
6.615.1 Detailed Description . . . . .	2971
6.615.2 Constructor & Destructor Documentation . . . . .	2971
6.615.2.1 ~PooledThreadListener . . . . .	2971
6.615.3 Member Function Documentation . . . . .	2971
6.615.3.1 onTaskCompleted . . . . .	2971
6.615.3.2 onTaskException . . . . .	2972
6.615.3.3 onTaskStarted . . . . .	2972
6.616decaf::net::PortUnreachableException Class Reference . . . . .	2973
6.616.1 Constructor & Destructor Documentation . . . . .	2973
6.616.1.1 PortUnreachableException . . . . .	2973
6.616.1.2 PortUnreachableException . . . . .	2973
6.616.1.3 PortUnreachableException . . . . .	2974
6.616.1.4 PortUnreachableException . . . . .	2974
6.616.1.5 PortUnreachableException . . . . .	2974
6.616.1.6 PortUnreachableException . . . . .	2974
6.616.1.7 ~PortUnreachableException . . . . .	2975
6.616.2 Member Function Documentation . . . . .	2975
6.616.2.1 clone . . . . .	2975
6.617activemq::core::PrefetchPolicy Class Reference . . . . .	2976
6.617.1 Detailed Description . . . . .	2977
6.617.2 Constructor & Destructor Documentation . . . . .	2977
6.617.2.1 PrefetchPolicy . . . . .	2977
6.617.2.2 ~PrefetchPolicy . . . . .	2977
6.617.3 Member Function Documentation . . . . .	2977
6.617.3.1 clone . . . . .	2977
6.617.3.2 configure . . . . .	2977
6.617.3.3 getDurableTopicPrefetch . . . . .	2978
6.617.3.4 getMaxPrefetchLimit . . . . .	2978
6.617.3.5 getQueueBrowserPrefetch . . . . .	2978

6.617.3.6	getQueuePrefetch . . . . .	2978
6.617.3.7	getTopicPrefetch . . . . .	2978
6.617.3.8	setDurableTopicPrefetch . . . . .	2979
6.617.3.9	setQueueBrowserPrefetch . . . . .	2979
6.617.3.10	setQueuePrefetch . . . . .	2979
6.617.3.11	setTopicPrefetch . . . . .	2979
6.618	activemq::util::PrimitiveList Class Reference . . . . .	2980
6.618.1	Detailed Description . . . . .	2982
6.618.2	Constructor & Destructor Documentation . . . . .	2982
6.618.2.1	PrimitiveList . . . . .	2982
6.618.2.2	~PrimitiveList . . . . .	2982
6.618.2.3	PrimitiveList . . . . .	2982
6.618.2.4	PrimitiveList . . . . .	2983
6.618.3	Member Function Documentation . . . . .	2983
6.618.3.1	getBool . . . . .	2983
6.618.3.2	getByte . . . . .	2983
6.618.3.3	getByteArray . . . . .	2984
6.618.3.4	getChar . . . . .	2984
6.618.3.5	getDouble . . . . .	2984
6.618.3.6	getFloat . . . . .	2985
6.618.3.7	getInt . . . . .	2985
6.618.3.8	getLong . . . . .	2985
6.618.3.9	getShort . . . . .	2986
6.618.3.10	getString . . . . .	2986
6.618.3.11	setBool . . . . .	2987
6.618.3.12	setByte . . . . .	2987
6.618.3.13	setByteArray . . . . .	2987
6.618.3.14	setChar . . . . .	2988
6.618.3.15	setDouble . . . . .	2988
6.618.3.16	setFloat . . . . .	2988
6.618.3.17	setInt . . . . .	2989
6.618.3.18	setLong . . . . .	2989
6.618.3.19	setShort . . . . .	2989
6.618.3.20	setString . . . . .	2990
6.618.3.21	toString . . . . .	2990
6.619	activemq::util::PrimitiveMap Class Reference . . . . .	2991

6.619.1 Detailed Description . . . . .	2993
6.619.2 Constructor & Destructor Documentation . . . . .	2993
6.619.2.1 PrimitiveMap . . . . .	2993
6.619.2.2 ~PrimitiveMap . . . . .	2993
6.619.2.3 PrimitiveMap . . . . .	2993
6.619.2.4 PrimitiveMap . . . . .	2993
6.619.3 Member Function Documentation . . . . .	2994
6.619.3.1 getBool . . . . .	2994
6.619.3.2 getByte . . . . .	2994
6.619.3.3 getByteArray . . . . .	2994
6.619.3.4 getChar . . . . .	2995
6.619.3.5 getDouble . . . . .	2995
6.619.3.6 getFloat . . . . .	2996
6.619.3.7 getInt . . . . .	2996
6.619.3.8 getLong . . . . .	2996
6.619.3.9 getShort . . . . .	2997
6.619.3.10 getString . . . . .	2997
6.619.3.11 setBool . . . . .	2998
6.619.3.12 setByte . . . . .	2998
6.619.3.13 setByteArray . . . . .	2998
6.619.3.14 setChar . . . . .	2998
6.619.3.15 setDouble . . . . .	2998
6.619.3.16 setFloat . . . . .	2999
6.619.3.17 setInt . . . . .	2999
6.619.3.18 setLong . . . . .	2999
6.619.3.19 setShort . . . . .	2999
6.619.3.20 setString . . . . .	3000
6.619.3.21 toString . . . . .	3000
6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference	3001
6.620.1 Detailed Description . . . . .	3002
6.620.2 Constructor & Destructor Documentation . . . . .	3003
6.620.2.1 PrimitiveTypesMarshaller . . . . .	3003
6.620.2.2 ~PrimitiveTypesMarshaller . . . . .	3003
6.620.3 Member Function Documentation . . . . .	3003
6.620.3.1 marshal . . . . .	3003
6.620.3.2 marshal . . . . .	3003

6.620.3.3 marshalList . . . . .	3003
6.620.3.4 marshalMap . . . . .	3004
6.620.3.5 marshalPrimitive . . . . .	3004
6.620.3.6 marshalPrimitiveList . . . . .	3004
6.620.3.7 marshalPrimitiveMap . . . . .	3005
6.620.3.8 unmarshal . . . . .	3005
6.620.3.9 unmarshal . . . . .	3005
6.620.3.10 unmarshalList . . . . .	3006
6.620.3.11 unmarshalMap . . . . .	3006
6.620.3.12 unmarshalPrimitive . . . . .	3006
6.620.3.13 unmarshalPrimitiveList . . . . .	3007
6.620.3.14 unmarshalPrimitiveMap . . . . .	3007
6.621 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference . . . . .	3008
6.621.1 Detailed Description . . . . .	3008
6.621.2 Field Documentation . . . . .	3009
6.621.2.1 boolValue . . . . .	3009
6.621.2.2 byteArrayValue . . . . .	3009
6.621.2.3 byteValue . . . . .	3009
6.621.2.4 charValue . . . . .	3009
6.621.2.5 doubleValue . . . . .	3009
6.621.2.6 floatValue . . . . .	3009
6.621.2.7 intValue . . . . .	3009
6.621.2.8 listValue . . . . .	3009
6.621.2.9 longValue . . . . .	3009
6.621.2.10 mapValue . . . . .	3009
6.621.2.11 shortValue . . . . .	3009
6.621.2.12 stringValue . . . . .	3009
6.622 activemq::util::PrimitiveValueConverter Class Reference . . . . .	3010
6.622.1 Detailed Description . . . . .	3010
6.622.2 Constructor & Destructor Documentation . . . . .	3010
6.622.2.1 PrimitiveValueConverter . . . . .	3010
6.622.2.2 ~PrimitiveValueConverter . . . . .	3010
6.622.3 Member Function Documentation . . . . .	3010
6.622.3.1 convert . . . . .	3010
6.623 activemq::util::PrimitiveValueNode Class Reference . . . . .	3012
6.623.1 Detailed Description . . . . .	3015

6.623.2 Member Enumeration Documentation . . . . .	3015
6.623.2.1 PrimitiveType . . . . .	3015
6.623.3 Constructor & Destructor Documentation . . . . .	3016
6.623.3.1 PrimitiveValueNode . . . . .	3016
6.623.3.2 PrimitiveValueNode . . . . .	3016
6.623.3.3 PrimitiveValueNode . . . . .	3016
6.623.3.4 PrimitiveValueNode . . . . .	3016
6.623.3.5 PrimitiveValueNode . . . . .	3017
6.623.3.6 PrimitiveValueNode . . . . .	3017
6.623.3.7 PrimitiveValueNode . . . . .	3017
6.623.3.8 PrimitiveValueNode . . . . .	3017
6.623.3.9 PrimitiveValueNode . . . . .	3017
6.623.3.10 PrimitiveValueNode . . . . .	3017
6.623.3.11 PrimitiveValueNode . . . . .	3018
6.623.3.12 PrimitiveValueNode . . . . .	3018
6.623.3.13 PrimitiveValueNode . . . . .	3018
6.623.3.14 PrimitiveValueNode . . . . .	3018
6.623.3.15 PrimitiveValueNode . . . . .	3018
6.623.3.16 ~PrimitiveValueNode . . . . .	3018
6.623.4 Member Function Documentation . . . . .	3018
6.623.4.1 clear . . . . .	3018
6.623.4.2 getBool . . . . .	3019
6.623.4.3 getByte . . . . .	3019
6.623.4.4 getByteArray . . . . .	3019
6.623.4.5 getChar . . . . .	3019
6.623.4.6 getDouble . . . . .	3020
6.623.4.7 getFloat . . . . .	3020
6.623.4.8 getInt . . . . .	3020
6.623.4.9 getList . . . . .	3020
6.623.4.10 getLong . . . . .	3021
6.623.4.11 getMap . . . . .	3021
6.623.4.12 getShort . . . . .	3021
6.623.4.13 getString . . . . .	3021
6.623.4.14 getType . . . . .	3022
6.623.4.15 getValue . . . . .	3022
6.623.4.16 operator= . . . . .	3022

6.623.4.17	operator==	3022
6.623.4.18	setBool	3022
6.623.4.19	setByte	3022
6.623.4.20	setByteArray	3023
6.623.4.21	setChar	3023
6.623.4.22	setDouble	3023
6.623.4.23	setFloat	3023
6.623.4.24	setInt	3023
6.623.4.25	setList	3023
6.623.4.26	setLong	3024
6.623.4.27	setMap	3024
6.623.4.28	setShort	3024
6.623.4.29	setString	3024
6.623.4.30	setValue	3024
6.623.4.31	toString	3025
6.624	decaf::security::Principal Class Reference	3026
6.624.1	Detailed Description	3026
6.624.2	Constructor & Destructor Documentation	3026
6.624.2.1	~Principal	3026
6.624.3	Member Function Documentation	3026
6.624.3.1	equals	3026
6.624.3.2	getName	3026
6.625	decaf::util::PriorityQueue< E > Class Template Reference	3028
6.625.1	Detailed Description	3029
6.625.2	Constructor & Destructor Documentation	3030
6.625.2.1	PriorityQueue	3030
6.625.2.2	PriorityQueue	3030
6.625.2.3	PriorityQueue	3030
6.625.2.4	PriorityQueue	3030
6.625.2.5	PriorityQueue	3031
6.625.2.6	~PriorityQueue	3031
6.625.3	Member Function Documentation	3031
6.625.3.1	add	3031
6.625.3.2	clear	3031
6.625.3.3	comparator	3032
6.625.3.4	iterator	3032

6.625.3.5 iterator . . . . .	3032
6.625.3.6 offer . . . . .	3032
6.625.3.7 operator= . . . . .	3033
6.625.3.8 operator= . . . . .	3033
6.625.3.9 peek . . . . .	3033
6.625.3.10 poll . . . . .	3033
6.625.3.11 remove . . . . .	3034
6.625.3.12 remove . . . . .	3034
6.625.3.13 size . . . . .	3034
6.625.4 Friends And Related Function Documentation . . . . .	3035
6.625.4.1 PriorityQueueIterator . . . . .	3035
6.626 activemq::commands::ProducerAck Class Reference . . . . .	3036
6.626.1 Constructor & Destructor Documentation . . . . .	3037
6.626.1.1 ProducerAck . . . . .	3037
6.626.1.2 ~ProducerAck . . . . .	3037
6.626.2 Member Function Documentation . . . . .	3037
6.626.2.1 cloneDataStructure . . . . .	3037
6.626.2.2 copyDataStructure . . . . .	3037
6.626.2.3 equals . . . . .	3037
6.626.2.4 getDataStructureType . . . . .	3037
6.626.2.5 getProducerId . . . . .	3038
6.626.2.6 getProducerId . . . . .	3038
6.626.2.7 getSize . . . . .	3038
6.626.2.8 isProducerAck . . . . .	3038
6.626.2.9 setProducerId . . . . .	3038
6.626.2.10 setSize . . . . .	3038
6.626.2.11 toString . . . . .	3038
6.626.2.12 visit . . . . .	3038
6.626.3 Field Documentation . . . . .	3039
6.626.3.1 ID_PRODUCERACK . . . . .	3039
6.626.3.2 producerId . . . . .	3039
6.626.3.3 size . . . . .	3039
6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference	3040
6.627.1 Detailed Description . . . . .	3040
6.627.2 Constructor & Destructor Documentation . . . . .	3041
6.627.2.1 ProducerAckMarshaller . . . . .	3041

6.627.2.2	~ProducerAckMarshaller . . . . .	3041
6.627.3	Member Function Documentation . . . . .	3041
6.627.3.1	createObject . . . . .	3041
6.627.3.2	getDataStructureType . . . . .	3041
6.627.3.3	looseMarshal . . . . .	3041
6.627.3.4	looseUnmarshal . . . . .	3042
6.627.3.5	tightMarshal1 . . . . .	3042
6.627.3.6	tightMarshal2 . . . . .	3042
6.627.3.7	tightUnmarshal . . . . .	3043
6.628	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference	3044
6.628.1	Detailed Description . . . . .	3044
6.628.2	Constructor & Destructor Documentation . . . . .	3045
6.628.2.1	ProducerAckMarshaller . . . . .	3045
6.628.2.2	~ProducerAckMarshaller . . . . .	3045
6.628.3	Member Function Documentation . . . . .	3045
6.628.3.1	createObject . . . . .	3045
6.628.3.2	getDataStructureType . . . . .	3045
6.628.3.3	looseMarshal . . . . .	3045
6.628.3.4	looseUnmarshal . . . . .	3046
6.628.3.5	tightMarshal1 . . . . .	3046
6.628.3.6	tightMarshal2 . . . . .	3046
6.628.3.7	tightUnmarshal . . . . .	3047
6.629	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference	3048
6.629.1	Detailed Description . . . . .	3048
6.629.2	Constructor & Destructor Documentation . . . . .	3049
6.629.2.1	ProducerAckMarshaller . . . . .	3049
6.629.2.2	~ProducerAckMarshaller . . . . .	3049
6.629.3	Member Function Documentation . . . . .	3049
6.629.3.1	createObject . . . . .	3049
6.629.3.2	getDataStructureType . . . . .	3049
6.629.3.3	looseMarshal . . . . .	3049
6.629.3.4	looseUnmarshal . . . . .	3050
6.629.3.5	tightMarshal1 . . . . .	3050
6.629.3.6	tightMarshal2 . . . . .	3050
6.629.3.7	tightUnmarshal . . . . .	3051
6.630	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference	3052



6.630.1 Detailed Description . . . . .	3052
6.630.2 Constructor & Destructor Documentation . . . . .	3053
6.630.2.1 ProducerAckMarshaller . . . . .	3053
6.630.2.2 ~ProducerAckMarshaller . . . . .	3053
6.630.3 Member Function Documentation . . . . .	3053
6.630.3.1 createObject . . . . .	3053
6.630.3.2 getDataStructureType . . . . .	3053
6.630.3.3 looseMarshal . . . . .	3053
6.630.3.4 looseUnmarshal . . . . .	3054
6.630.3.5 tightMarshal1 . . . . .	3054
6.630.3.6 tightMarshal2 . . . . .	3054
6.630.3.7 tightUnmarshal . . . . .	3055
6.631activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller Class Reference	3056
6.631.1 Detailed Description . . . . .	3056
6.631.2 Constructor & Destructor Documentation . . . . .	3057
6.631.2.1 ProducerAckMarshaller . . . . .	3057
6.631.2.2 ~ProducerAckMarshaller . . . . .	3057
6.631.3 Member Function Documentation . . . . .	3057
6.631.3.1 createObject . . . . .	3057
6.631.3.2 getDataStructureType . . . . .	3057
6.631.3.3 looseMarshal . . . . .	3057
6.631.3.4 looseUnmarshal . . . . .	3058
6.631.3.5 tightMarshal1 . . . . .	3058
6.631.3.6 tightMarshal2 . . . . .	3058
6.631.3.7 tightUnmarshal . . . . .	3059
6.632activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference	3060
6.632.1 Detailed Description . . . . .	3060
6.632.2 Constructor & Destructor Documentation . . . . .	3061
6.632.2.1 ProducerAckMarshaller . . . . .	3061
6.632.2.2 ~ProducerAckMarshaller . . . . .	3061
6.632.3 Member Function Documentation . . . . .	3061
6.632.3.1 createObject . . . . .	3061
6.632.3.2 getDataStructureType . . . . .	3061
6.632.3.3 looseMarshal . . . . .	3061
6.632.3.4 looseUnmarshal . . . . .	3062
6.632.3.5 tightMarshal1 . . . . .	3062

6.632.3.6 tightMarshal2 . . . . .	3062
6.632.3.7 tightUnmarshal . . . . .	3063
6.633activemq::cmsutil::ProducerCallback Class Reference . . . . .	3064
6.633.1 Detailed Description . . . . .	3064
6.633.2 Constructor & Destructor Documentation . . . . .	3064
6.633.2.1 ~ProducerCallback . . . . .	3064
6.633.3 Member Function Documentation . . . . .	3064
6.633.3.1 doInCms . . . . .	3064
6.634activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference . . . . .	3065
6.634.1 Constructor & Destructor Documentation . . . . .	3065
6.634.1.1 ProducerExecutor . . . . .	3065
6.634.1.2 ProducerExecutor . . . . .	3065
6.634.1.3 ~ProducerExecutor . . . . .	3065
6.634.2 Member Function Documentation . . . . .	3065
6.634.2.1 doInCms . . . . .	3065
6.634.2.2 getDestination . . . . .	3066
6.634.2.3 operator= . . . . .	3066
6.634.3 Field Documentation . . . . .	3066
6.634.3.1 action . . . . .	3066
6.634.3.2 destination . . . . .	3066
6.634.3.3 parent . . . . .	3066
6.635activemq::commands::ProducerId Class Reference . . . . .	3067
6.635.1 Member Typedef Documentation . . . . .	3068
6.635.1.1 COMPARATOR . . . . .	3068
6.635.2 Constructor & Destructor Documentation . . . . .	3068
6.635.2.1 ProducerId . . . . .	3068
6.635.2.2 ProducerId . . . . .	3068
6.635.2.3 ProducerId . . . . .	3068
6.635.2.4 ProducerId . . . . .	3068
6.635.2.5 ~ProducerId . . . . .	3068
6.635.3 Member Function Documentation . . . . .	3068
6.635.3.1 cloneDataStructure . . . . .	3068
6.635.3.2 compareTo . . . . .	3068
6.635.3.3 copyDataStructure . . . . .	3068
6.635.3.4 equals . . . . .	3069
6.635.3.5 equals . . . . .	3069

6.635.3.6	getConnectionId . . . . .	3069
6.635.3.7	getConnectionId . . . . .	3069
6.635.3.8	getDataStructureType . . . . .	3069
6.635.3.9	getParentId . . . . .	3070
6.635.3.10	getSessionId . . . . .	3070
6.635.3.11	getValue . . . . .	3070
6.635.3.12	operator< . . . . .	3070
6.635.3.13	operator= . . . . .	3070
6.635.3.14	operator== . . . . .	3070
6.635.3.15	setConnectionId . . . . .	3070
6.635.3.16	setProducerSessionKey . . . . .	3070
6.635.3.17	setSessionId . . . . .	3070
6.635.3.18	setValue . . . . .	3070
6.635.3.19	toString . . . . .	3070
6.635.4	Field Documentation . . . . .	3070
6.635.4.1	connectionId . . . . .	3070
6.635.4.2	ID_PRODUCERID . . . . .	3070
6.635.4.3	sessionId . . . . .	3071
6.635.4.4	value . . . . .	3071
6.636	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference	3072
6.636.1	Detailed Description . . . . .	3072
6.636.2	Constructor & Destructor Documentation . . . . .	3073
6.636.2.1	ProducerIdMarshaller . . . . .	3073
6.636.2.2	~ProducerIdMarshaller . . . . .	3073
6.636.3	Member Function Documentation . . . . .	3073
6.636.3.1	createObject . . . . .	3073
6.636.3.2	getDataStructureType . . . . .	3073
6.636.3.3	looseMarshal . . . . .	3073
6.636.3.4	looseUnmarshal . . . . .	3074
6.636.3.5	tightMarshal1 . . . . .	3074
6.636.3.6	tightMarshal2 . . . . .	3074
6.636.3.7	tightUnmarshal . . . . .	3075
6.637	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference	3076
6.637.1	Detailed Description . . . . .	3076
6.637.2	Constructor & Destructor Documentation . . . . .	3077
6.637.2.1	ProducerIdMarshaller . . . . .	3077

6.637.2.2 ~ProducerIdMarshaller . . . . .	3077
6.637.3 Member Function Documentation . . . . .	3077
6.637.3.1 createObject . . . . .	3077
6.637.3.2 getDataStructureType . . . . .	3077
6.637.3.3 looseMarshal . . . . .	3077
6.637.3.4 looseUnmarshal . . . . .	3078
6.637.3.5 tightMarshal1 . . . . .	3078
6.637.3.6 tightMarshal2 . . . . .	3078
6.637.3.7 tightUnmarshal . . . . .	3079
6.638activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference	3080
6.638.1 Detailed Description . . . . .	3080
6.638.2 Constructor & Destructor Documentation . . . . .	3081
6.638.2.1 ProducerIdMarshaller . . . . .	3081
6.638.2.2 ~ProducerIdMarshaller . . . . .	3081
6.638.3 Member Function Documentation . . . . .	3081
6.638.3.1 createObject . . . . .	3081
6.638.3.2 getDataStructureType . . . . .	3081
6.638.3.3 looseMarshal . . . . .	3081
6.638.3.4 looseUnmarshal . . . . .	3082
6.638.3.5 tightMarshal1 . . . . .	3082
6.638.3.6 tightMarshal2 . . . . .	3082
6.638.3.7 tightUnmarshal . . . . .	3083
6.639activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference	3084
6.639.1 Detailed Description . . . . .	3084
6.639.2 Constructor & Destructor Documentation . . . . .	3085
6.639.2.1 ProducerIdMarshaller . . . . .	3085
6.639.2.2 ~ProducerIdMarshaller . . . . .	3085
6.639.3 Member Function Documentation . . . . .	3085
6.639.3.1 createObject . . . . .	3085
6.639.3.2 getDataStructureType . . . . .	3085
6.639.3.3 looseMarshal . . . . .	3085
6.639.3.4 looseUnmarshal . . . . .	3086
6.639.3.5 tightMarshal1 . . . . .	3086
6.639.3.6 tightMarshal2 . . . . .	3086
6.639.3.7 tightUnmarshal . . . . .	3087
6.640activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class Reference	3088

6.640.1 Detailed Description . . . . .	3088
6.640.2 Constructor & Destructor Documentation . . . . .	3089
6.640.2.1 ProducerIdMarshaller . . . . .	3089
6.640.2.2 ~ProducerIdMarshaller . . . . .	3089
6.640.3 Member Function Documentation . . . . .	3089
6.640.3.1 createObject . . . . .	3089
6.640.3.2 getDataStructureType . . . . .	3089
6.640.3.3 looseMarshal . . . . .	3089
6.640.3.4 looseUnmarshal . . . . .	3090
6.640.3.5 tightMarshal1 . . . . .	3090
6.640.3.6 tightMarshal2 . . . . .	3090
6.640.3.7 tightUnmarshal . . . . .	3091
6.641activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference	3092
6.641.1 Detailed Description . . . . .	3092
6.641.2 Constructor & Destructor Documentation . . . . .	3093
6.641.2.1 ProducerIdMarshaller . . . . .	3093
6.641.2.2 ~ProducerIdMarshaller . . . . .	3093
6.641.3 Member Function Documentation . . . . .	3093
6.641.3.1 createObject . . . . .	3093
6.641.3.2 getDataStructureType . . . . .	3093
6.641.3.3 looseMarshal . . . . .	3093
6.641.3.4 looseUnmarshal . . . . .	3094
6.641.3.5 tightMarshal1 . . . . .	3094
6.641.3.6 tightMarshal2 . . . . .	3094
6.641.3.7 tightUnmarshal . . . . .	3095
6.642activemq::commands::ProducerInfo Class Reference . . . . .	3096
6.642.1 Constructor & Destructor Documentation . . . . .	3097
6.642.1.1 ProducerInfo . . . . .	3097
6.642.1.2 ~ProducerInfo . . . . .	3097
6.642.2 Member Function Documentation . . . . .	3097
6.642.2.1 cloneDataStructure . . . . .	3097
6.642.2.2 copyDataStructure . . . . .	3097
6.642.2.3 createRemoveCommand . . . . .	3097
6.642.2.4 equals . . . . .	3097
6.642.2.5 getBrokerPath . . . . .	3098
6.642.2.6 getBrokerPath . . . . .	3098

6.642.2.7	getDataStructureType . . . . .	3098
6.642.2.8	getDestination . . . . .	3098
6.642.2.9	getDestination . . . . .	3098
6.642.2.10	getProducerId . . . . .	3098
6.642.2.11	getProducerId . . . . .	3098
6.642.2.12	getWindowSize . . . . .	3098
6.642.2.13	sDispatchAsync . . . . .	3098
6.642.2.14	sProducerInfo . . . . .	3098
6.642.2.15	setBrokerPath . . . . .	3099
6.642.2.16	setDestination . . . . .	3099
6.642.2.17	setDispatchAsync . . . . .	3099
6.642.2.18	setProducerId . . . . .	3099
6.642.2.19	setWindowSize . . . . .	3099
6.642.2.20	toString . . . . .	3099
6.642.2.21	visit . . . . .	3099
6.642.3	Field Documentation . . . . .	3100
6.642.3.1	brokerPath . . . . .	3100
6.642.3.2	destination . . . . .	3100
6.642.3.3	dispatchAsync . . . . .	3100
6.642.3.4	ID_PRODUCERINFO . . . . .	3100
6.642.3.5	producerId . . . . .	3100
6.642.3.6	windowSize . . . . .	3100
6.643	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference . . . . .	3101
6.643.1	Detailed Description . . . . .	3101
6.643.2	Constructor & Destructor Documentation . . . . .	3102
6.643.2.1	ProducerInfoMarshaller . . . . .	3102
6.643.2.2	~ProducerInfoMarshaller . . . . .	3102
6.643.3	Member Function Documentation . . . . .	3102
6.643.3.1	createObject . . . . .	3102
6.643.3.2	getDataStructureType . . . . .	3102
6.643.3.3	looseMarshal . . . . .	3102
6.643.3.4	looseUnmarshal . . . . .	3103
6.643.3.5	tightMarshal1 . . . . .	3103
6.643.3.6	tightMarshal2 . . . . .	3103
6.643.3.7	tightUnmarshal . . . . .	3104

6.644activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference . . . . .	3105
6.644.1 Detailed Description . . . . .	3105
6.644.2 Constructor & Destructor Documentation . . . . .	3106
6.644.2.1 ProducerInfoMarshaller . . . . .	3106
6.644.2.2 ~ProducerInfoMarshaller . . . . .	3106
6.644.3 Member Function Documentation . . . . .	3106
6.644.3.1 createObject . . . . .	3106
6.644.3.2 getDataStructureType . . . . .	3106
6.644.3.3 looseMarshal . . . . .	3106
6.644.3.4 looseUnmarshal . . . . .	3107
6.644.3.5 tightMarshal1 . . . . .	3107
6.644.3.6 tightMarshal2 . . . . .	3107
6.644.3.7 tightUnmarshal . . . . .	3108
6.645activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference . . . . .	3109
6.645.1 Detailed Description . . . . .	3109
6.645.2 Constructor & Destructor Documentation . . . . .	3110
6.645.2.1 ProducerInfoMarshaller . . . . .	3110
6.645.2.2 ~ProducerInfoMarshaller . . . . .	3110
6.645.3 Member Function Documentation . . . . .	3110
6.645.3.1 createObject . . . . .	3110
6.645.3.2 getDataStructureType . . . . .	3110
6.645.3.3 looseMarshal . . . . .	3110
6.645.3.4 looseUnmarshal . . . . .	3111
6.645.3.5 tightMarshal1 . . . . .	3111
6.645.3.6 tightMarshal2 . . . . .	3111
6.645.3.7 tightUnmarshal . . . . .	3112
6.646activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference . . . . .	3113
6.646.1 Detailed Description . . . . .	3113
6.646.2 Constructor & Destructor Documentation . . . . .	3114
6.646.2.1 ProducerInfoMarshaller . . . . .	3114
6.646.2.2 ~ProducerInfoMarshaller . . . . .	3114
6.646.3 Member Function Documentation . . . . .	3114
6.646.3.1 createObject . . . . .	3114
6.646.3.2 getDataStructureType . . . . .	3114

6.646.3.3 looseMarshal . . . . .	3114
6.646.3.4 looseUnmarshal . . . . .	3115
6.646.3.5 tightMarshal1 . . . . .	3115
6.646.3.6 tightMarshal2 . . . . .	3115
6.646.3.7 tightUnmarshal . . . . .	3116
6.647activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference . . . . .	3117
6.647.1 Detailed Description . . . . .	3117
6.647.2 Constructor & Destructor Documentation . . . . .	3118
6.647.2.1 ProducerInfoMarshaller . . . . .	3118
6.647.2.2 ~ProducerInfoMarshaller . . . . .	3118
6.647.3 Member Function Documentation . . . . .	3118
6.647.3.1 createObject . . . . .	3118
6.647.3.2 getDataStructureType . . . . .	3118
6.647.3.3 looseMarshal . . . . .	3118
6.647.3.4 looseUnmarshal . . . . .	3119
6.647.3.5 tightMarshal1 . . . . .	3119
6.647.3.6 tightMarshal2 . . . . .	3119
6.647.3.7 tightUnmarshal . . . . .	3120
6.648activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference . . . . .	3121
6.648.1 Detailed Description . . . . .	3121
6.648.2 Constructor & Destructor Documentation . . . . .	3122
6.648.2.1 ProducerInfoMarshaller . . . . .	3122
6.648.2.2 ~ProducerInfoMarshaller . . . . .	3122
6.648.3 Member Function Documentation . . . . .	3122
6.648.3.1 createObject . . . . .	3122
6.648.3.2 getDataStructureType . . . . .	3122
6.648.3.3 looseMarshal . . . . .	3122
6.648.3.4 looseUnmarshal . . . . .	3123
6.648.3.5 tightMarshal1 . . . . .	3123
6.648.3.6 tightMarshal2 . . . . .	3123
6.648.3.7 tightUnmarshal . . . . .	3124
6.649activemq::state::ProducerState Class Reference . . . . .	3125
6.649.1 Constructor & Destructor Documentation . . . . .	3125
6.649.1.1 ProducerState . . . . .	3125
6.649.1.2 ~ProducerState . . . . .	3125



6.649.2 Member Function Documentation . . . . .	3125
6.649.2.1 getInfo . . . . .	3125
6.649.2.2 getTransactionState . . . . .	3125
6.649.2.3 setTransactionState . . . . .	3125
6.649.2.4 toString . . . . .	3125
6.650decaf::util::Properties Class Reference . . . . .	3126
6.650.1 Detailed Description . . . . .	3127
6.650.2 Constructor & Destructor Documentation . . . . .	3128
6.650.2.1 Properties . . . . .	3128
6.650.2.2 Properties . . . . .	3128
6.650.2.3 ~Properties . . . . .	3128
6.650.3 Member Function Documentation . . . . .	3128
6.650.3.1 clear . . . . .	3128
6.650.3.2 clone . . . . .	3128
6.650.3.3 copy . . . . .	3128
6.650.3.4 equals . . . . .	3128
6.650.3.5 getProperty . . . . .	3128
6.650.3.6 getProperty . . . . .	3129
6.650.3.7 hasProperty . . . . .	3129
6.650.3.8 isEmpty . . . . .	3129
6.650.3.9 load . . . . .	3129
6.650.3.10load . . . . .	3131
6.650.3.11operator= . . . . .	3132
6.650.3.12propertyName . . . . .	3132
6.650.3.13remove . . . . .	3132
6.650.3.14setProperty . . . . .	3132
6.650.3.15size . . . . .	3132
6.650.3.16store . . . . .	3133
6.650.3.17store . . . . .	3133
6.650.3.18oArray . . . . .	3134
6.650.3.19oString . . . . .	3134
6.650.4 Field Documentation . . . . .	3134
6.650.4.1 defaults . . . . .	3134
6.651decaf::util::logging::PropertiesChangeListener Class Reference . . . . .	3135
6.651.1 Detailed Description . . . . .	3135
6.651.2 Constructor & Destructor Documentation . . . . .	3135

6.651.2.1 ~PropertiesChangeListener . . . . .	3135
6.651.3 Member Function Documentation . . . . .	3135
6.651.3.1 onPropertiesReset . . . . .	3135
6.651.3.2 onPropertyChanged . . . . .	3135
6.652decaf::net::ProtocolException Class Reference . . . . .	3137
6.652.1 Constructor & Destructor Documentation . . . . .	3137
6.652.1.1 ProtocolException . . . . .	3137
6.652.1.2 ProtocolException . . . . .	3137
6.652.1.3 ProtocolException . . . . .	3138
6.652.1.4 ProtocolException . . . . .	3138
6.652.1.5 ProtocolException . . . . .	3138
6.652.1.6 ProtocolException . . . . .	3138
6.652.1.7 ~ProtocolException . . . . .	3139
6.652.2 Member Function Documentation . . . . .	3139
6.652.2.1 clone . . . . .	3139
6.653decaf::security::PublicKey Class Reference . . . . .	3140
6.653.1 Detailed Description . . . . .	3140
6.653.2 Constructor & Destructor Documentation . . . . .	3140
6.653.2.1 ~PublicKey . . . . .	3140
6.654decaf::io::PushbackInputStream Class Reference . . . . .	3141
6.654.1 Detailed Description . . . . .	3142
6.654.2 Constructor & Destructor Documentation . . . . .	3143
6.654.2.1 PushbackInputStream . . . . .	3143
6.654.2.2 PushbackInputStream . . . . .	3143
6.654.2.3 ~PushbackInputStream . . . . .	3143
6.654.3 Member Function Documentation . . . . .	3143
6.654.3.1 available . . . . .	3143
6.654.3.2 doReadArrayBounded . . . . .	3144
6.654.3.3 doReadByte . . . . .	3144
6.654.3.4 mark . . . . .	3144
6.654.3.5 markSupported . . . . .	3144
6.654.3.6 reset . . . . .	3145
6.654.3.7 skip . . . . .	3145
6.654.3.8 unread . . . . .	3146
6.654.3.9 unread . . . . .	3146
6.654.3.10unread . . . . .	3147

6.655cms::Queue Class Reference . . . . .	3148
6.655.1 Detailed Description . . . . .	3148
6.655.2 Constructor & Destructor Documentation . . . . .	3148
6.655.2.1 ~Queue . . . . .	3148
6.655.3 Member Function Documentation . . . . .	3148
6.655.3.1 getQueueName . . . . .	3148
6.656decaf::util::Queue< E > Class Template Reference . . . . .	3149
6.656.1 Detailed Description . . . . .	3149
6.656.2 Constructor & Destructor Documentation . . . . .	3150
6.656.2.1 ~Queue . . . . .	3150
6.656.3 Member Function Documentation . . . . .	3150
6.656.3.1 element . . . . .	3150
6.656.3.2 offer . . . . .	3150
6.656.3.3 peek . . . . .	3151
6.656.3.4 poll . . . . .	3151
6.656.3.5 remove . . . . .	3151
6.657cms::QueueBrowser Class Reference . . . . .	3153
6.657.1 Detailed Description . . . . .	3153
6.657.2 Constructor & Destructor Documentation . . . . .	3153
6.657.2.1 ~QueueBrowser . . . . .	3153
6.657.3 Member Function Documentation . . . . .	3153
6.657.3.1 getEnumeration . . . . .	3153
6.657.3.2 getMessageSelector . . . . .	3154
6.657.3.3 getQueue . . . . .	3154
6.658decaf::util::Random Class Reference . . . . .	3155
6.658.1 Detailed Description . . . . .	3156
6.658.2 Constructor & Destructor Documentation . . . . .	3156
6.658.2.1 Random . . . . .	3156
6.658.2.2 Random . . . . .	3156
6.658.3 Member Function Documentation . . . . .	3156
6.658.3.1 next . . . . .	3156
6.658.3.2 nextBoolean . . . . .	3157
6.658.3.3 nextBytes . . . . .	3157
6.658.3.4 nextBytes . . . . .	3157
6.658.3.5 nextDouble . . . . .	3158
6.658.3.6 nextFloat . . . . .	3158

6.658.3.7	nextGaussian . . . . .	3158
6.658.3.8	nextInt . . . . .	3158
6.658.3.9	nextInt . . . . .	3159
6.658.3.10	nextLong . . . . .	3159
6.658.3.11	setSeed . . . . .	3159
6.659	decaf::lang::Readable Class Reference . . . . .	3160
6.659.1	Detailed Description . . . . .	3160
6.659.2	Constructor & Destructor Documentation . . . . .	3160
6.659.2.1	~Readable . . . . .	3160
6.659.3	Member Function Documentation . . . . .	3160
6.659.3.1	read . . . . .	3160
6.660	activemq::transport::inactivity::ReadChecker Class Reference . . . . .	3162
6.660.1	Detailed Description . . . . .	3162
6.660.2	Constructor & Destructor Documentation . . . . .	3162
6.660.2.1	ReadChecker . . . . .	3162
6.660.2.2	~ReadChecker . . . . .	3162
6.660.3	Member Function Documentation . . . . .	3162
6.660.3.1	run . . . . .	3162
6.661	decaf::io::Reader Class Reference . . . . .	3163
6.661.1	Constructor & Destructor Documentation . . . . .	3164
6.661.1.1	Reader . . . . .	3164
6.661.1.2	~Reader . . . . .	3164
6.661.2	Member Function Documentation . . . . .	3164
6.661.2.1	doReadArray . . . . .	3164
6.661.2.2	doReadArrayBounded . . . . .	3164
6.661.2.3	doReadChar . . . . .	3165
6.661.2.4	doReadCharBuffer . . . . .	3165
6.661.2.5	doReadVector . . . . .	3165
6.661.2.6	mark . . . . .	3165
6.661.2.7	markSupported . . . . .	3165
6.661.2.8	read . . . . .	3165
6.661.2.9	read . . . . .	3166
6.661.2.10	read . . . . .	3166
6.661.2.11	read . . . . .	3167
6.661.2.12	read . . . . .	3167
6.661.2.13	ready . . . . .	3167

6.661.2.14	reset . . . . .	3168
6.661.2.15	skip . . . . .	3168
6.662	decaf::nio::ReadOnlyBufferException Class Reference . . . . .	3169
6.662.1	Constructor & Destructor Documentation . . . . .	3169
6.662.1.1	ReadOnlyBufferException . . . . .	3169
6.662.1.2	ReadOnlyBufferException . . . . .	3169
6.662.1.3	ReadOnlyBufferException . . . . .	3170
6.662.1.4	ReadOnlyBufferException . . . . .	3170
6.662.1.5	ReadOnlyBufferException . . . . .	3170
6.662.1.6	ReadOnlyBufferException . . . . .	3170
6.662.1.7	~ReadOnlyBufferException . . . . .	3171
6.662.2	Member Function Documentation . . . . .	3171
6.662.2.1	clone . . . . .	3171
6.663	decaf::util::concurrent::locks::ReadWriteLock Class Reference . . . . .	3172
6.663.1	Detailed Description . . . . .	3172
6.663.2	Constructor & Destructor Documentation . . . . .	3173
6.663.2.1	~ReadWriteLock . . . . .	3173
6.663.3	Member Function Documentation . . . . .	3173
6.663.3.1	readLock . . . . .	3173
6.663.3.2	writeLock . . . . .	3173
6.664	activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference . . . . .	3174
6.664.1	Constructor & Destructor Documentation . . . . .	3175
6.664.1.1	ReceiveExecutor . . . . .	3175
6.664.1.2	ReceiveExecutor . . . . .	3175
6.664.1.3	~ReceiveExecutor . . . . .	3175
6.664.2	Member Function Documentation . . . . .	3175
6.664.2.1	doInCms . . . . .	3175
6.664.2.2	getDestination . . . . .	3175
6.664.2.3	getMessage . . . . .	3175
6.664.2.4	operator= . . . . .	3175
6.664.3	Field Documentation . . . . .	3176
6.664.3.1	destination . . . . .	3176
6.664.3.2	message . . . . .	3176
6.664.3.3	noLocal . . . . .	3176
6.664.3.4	parent . . . . .	3176
6.664.3.5	selector . . . . .	3176

6.665	activemq::core::RedeliveryPolicy Class Reference . . . . .	3177
6.665.1	Detailed Description . . . . .	3178
6.665.2	Constructor & Destructor Documentation . . . . .	3178
6.665.2.1	RedeliveryPolicy . . . . .	3178
6.665.2.2	~RedeliveryPolicy . . . . .	3178
6.665.3	Member Function Documentation . . . . .	3178
6.665.3.1	clone . . . . .	3178
6.665.3.2	configure . . . . .	3178
6.665.3.3	getBackOffMultiplier . . . . .	3179
6.665.3.4	getCollisionAvoidancePercent . . . . .	3179
6.665.3.5	getInitialRedeliveryDelay . . . . .	3179
6.665.3.6	getMaximumRedeliveries . . . . .	3179
6.665.3.7	getRedeliveryDelay . . . . .	3179
6.665.3.8	isUseCollisionAvoidance . . . . .	3180
6.665.3.9	isUseExponentialBackOff . . . . .	3180
6.665.3.10	setBackOffMultiplier . . . . .	3180
6.665.3.11	setCollisionAvoidancePercent . . . . .	3180
6.665.3.12	setInitialRedeliveryDelay . . . . .	3180
6.665.3.13	setMaximumRedeliveries . . . . .	3181
6.665.3.14	setUseCollisionAvoidance . . . . .	3181
6.665.3.15	setUseExponentialBackOff . . . . .	3181
6.665.4	Field Documentation . . . . .	3181
6.665.4.1	NO_MAXIMUM_REDELIVERIES . . . . .	3181
6.666	decaf::util::concurrent::locks::ReentrantLock Class Reference . . . . .	3182
6.666.1	Detailed Description . . . . .	3183
6.666.2	Constructor & Destructor Documentation . . . . .	3183
6.666.2.1	ReentrantLock . . . . .	3183
6.666.2.2	~ReentrantLock . . . . .	3183
6.666.3	Member Function Documentation . . . . .	3183
6.666.3.1	getHoldCount . . . . .	3183
6.666.3.2	isFair . . . . .	3184
6.666.3.3	isHeldByCurrentThread . . . . .	3184
6.666.3.4	isLocked . . . . .	3184
6.666.3.5	lock . . . . .	3185
6.666.3.6	lockInterruptibly . . . . .	3185
6.666.3.7	newCondition . . . . .	3186

6.666.3.8 toString . . . . .	3186
6.666.3.9 tryLock . . . . .	3186
6.666.3.10 tryLock . . . . .	3187
6.666.3.11 unlock . . . . .	3188
6.667 decaf::util::concurrent::RejectedExecutionException Class Reference . . . . .	3189
6.667.1 Constructor & Destructor Documentation . . . . .	3189
6.667.1.1 RejectedExecutionException . . . . .	3189
6.667.1.2 RejectedExecutionException . . . . .	3189
6.667.1.3 RejectedExecutionException . . . . .	3190
6.667.1.4 RejectedExecutionException . . . . .	3190
6.667.1.5 RejectedExecutionException . . . . .	3190
6.667.1.6 RejectedExecutionException . . . . .	3190
6.667.1.7 ~RejectedExecutionException . . . . .	3191
6.667.2 Member Function Documentation . . . . .	3191
6.667.2.1 clone . . . . .	3191
6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference . . . . .	3192
6.668.1 Detailed Description . . . . .	3192
6.668.2 Constructor & Destructor Documentation . . . . .	3192
6.668.2.1 ~RejectedExecutionHandler . . . . .	3192
6.668.3 Member Function Documentation . . . . .	3192
6.668.3.1 rejectedExecution . . . . .	3192
6.669 activemq::commands::RemoveInfo Class Reference . . . . .	3194
6.669.1 Constructor & Destructor Documentation . . . . .	3195
6.669.1.1 RemoveInfo . . . . .	3195
6.669.1.2 ~RemoveInfo . . . . .	3195
6.669.2 Member Function Documentation . . . . .	3195
6.669.2.1 cloneDataStructure . . . . .	3195
6.669.2.2 copyDataStructure . . . . .	3195
6.669.2.3 equals . . . . .	3195
6.669.2.4 getDataStructureType . . . . .	3195
6.669.2.5 getLastDeliveredSequenceId . . . . .	3196
6.669.2.6 getObjectId . . . . .	3196
6.669.2.7 getObjectId . . . . .	3196
6.669.2.8 isRemoveInfo . . . . .	3196
6.669.2.9 setLastDeliveredSequenceId . . . . .	3196
6.669.2.10 setObjectId . . . . .	3196

6.669.2.11toString . . . . .	3196
6.669.2.12visit . . . . .	3196
6.669.3 Field Documentation . . . . .	3197
6.669.3.1 ID_REMOVEINFO . . . . .	3197
6.669.3.2 lastDeliveredSequenceId . . . . .	3197
6.669.3.3 objectId . . . . .	3197
6.670activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference	3198
6.670.1 Detailed Description . . . . .	3198
6.670.2 Constructor & Destructor Documentation . . . . .	3199
6.670.2.1 RemoveInfoMarshaller . . . . .	3199
6.670.2.2 ~RemoveInfoMarshaller . . . . .	3199
6.670.3 Member Function Documentation . . . . .	3199
6.670.3.1 createObject . . . . .	3199
6.670.3.2 getDataStructureType . . . . .	3199
6.670.3.3 looseMarshal . . . . .	3199
6.670.3.4 looseUnmarshal . . . . .	3200
6.670.3.5 tightMarshal1 . . . . .	3200
6.670.3.6 tightMarshal2 . . . . .	3200
6.670.3.7 tightUnmarshal . . . . .	3201
6.671activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class Reference	3202
6.671.1 Detailed Description . . . . .	3202
6.671.2 Constructor & Destructor Documentation . . . . .	3203
6.671.2.1 RemoveInfoMarshaller . . . . .	3203
6.671.2.2 ~RemoveInfoMarshaller . . . . .	3203
6.671.3 Member Function Documentation . . . . .	3203
6.671.3.1 createObject . . . . .	3203
6.671.3.2 getDataStructureType . . . . .	3203
6.671.3.3 looseMarshal . . . . .	3203
6.671.3.4 looseUnmarshal . . . . .	3204
6.671.3.5 tightMarshal1 . . . . .	3204
6.671.3.6 tightMarshal2 . . . . .	3204
6.671.3.7 tightUnmarshal . . . . .	3205
6.672activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference	3206
6.672.1 Detailed Description . . . . .	3206
6.672.2 Constructor & Destructor Documentation . . . . .	3207
6.672.2.1 RemoveInfoMarshaller . . . . .	3207



6.672.2.2 ~RemoveInfoMarshaller . . . . .	3207
6.672.3 Member Function Documentation . . . . .	3207
6.672.3.1 createObject . . . . .	3207
6.672.3.2 getDataStructureType . . . . .	3207
6.672.3.3 looseMarshal . . . . .	3207
6.672.3.4 looseUnmarshal . . . . .	3208
6.672.3.5 tightMarshal1 . . . . .	3208
6.672.3.6 tightMarshal2 . . . . .	3208
6.672.3.7 tightUnmarshal . . . . .	3209
6.673activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference	3210
6.673.1 Detailed Description . . . . .	3210
6.673.2 Constructor & Destructor Documentation . . . . .	3211
6.673.2.1 RemoveInfoMarshaller . . . . .	3211
6.673.2.2 ~RemoveInfoMarshaller . . . . .	3211
6.673.3 Member Function Documentation . . . . .	3211
6.673.3.1 createObject . . . . .	3211
6.673.3.2 getDataStructureType . . . . .	3211
6.673.3.3 looseMarshal . . . . .	3211
6.673.3.4 looseUnmarshal . . . . .	3212
6.673.3.5 tightMarshal1 . . . . .	3212
6.673.3.6 tightMarshal2 . . . . .	3212
6.673.3.7 tightUnmarshal . . . . .	3213
6.674activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference	3214
6.674.1 Detailed Description . . . . .	3214
6.674.2 Constructor & Destructor Documentation . . . . .	3215
6.674.2.1 RemoveInfoMarshaller . . . . .	3215
6.674.2.2 ~RemoveInfoMarshaller . . . . .	3215
6.674.3 Member Function Documentation . . . . .	3215
6.674.3.1 createObject . . . . .	3215
6.674.3.2 getDataStructureType . . . . .	3215
6.674.3.3 looseMarshal . . . . .	3215
6.674.3.4 looseUnmarshal . . . . .	3216
6.674.3.5 tightMarshal1 . . . . .	3216
6.674.3.6 tightMarshal2 . . . . .	3216
6.674.3.7 tightUnmarshal . . . . .	3217
6.675activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference	3218

6.675.1 Detailed Description . . . . .	3218
6.675.2 Constructor & Destructor Documentation . . . . .	3219
6.675.2.1 RemoveInfoMarshaller . . . . .	3219
6.675.2.2 ~RemoveInfoMarshaller . . . . .	3219
6.675.3 Member Function Documentation . . . . .	3219
6.675.3.1 createObject . . . . .	3219
6.675.3.2 getDataStructureType . . . . .	3219
6.675.3.3 looseMarshal . . . . .	3219
6.675.3.4 looseUnmarshal . . . . .	3220
6.675.3.5 tightMarshal1 . . . . .	3220
6.675.3.6 tightMarshal2 . . . . .	3220
6.675.3.7 tightUnmarshal . . . . .	3221
6.676activemq::commands::RemoveSubscriptionInfo Class Reference . . . . .	3222
6.676.1 Constructor & Destructor Documentation . . . . .	3223
6.676.1.1 RemoveSubscriptionInfo . . . . .	3223
6.676.1.2 ~RemoveSubscriptionInfo . . . . .	3223
6.676.2 Member Function Documentation . . . . .	3223
6.676.2.1 cloneDataStructure . . . . .	3223
6.676.2.2 copyDataStructure . . . . .	3223
6.676.2.3 equals . . . . .	3223
6.676.2.4 getClientId . . . . .	3224
6.676.2.5 getClientId . . . . .	3224
6.676.2.6 getConnectionId . . . . .	3224
6.676.2.7 getConnectionId . . . . .	3224
6.676.2.8 getDataStructureType . . . . .	3224
6.676.2.9 getSubscriptionName . . . . .	3224
6.676.2.10getSubscriptionName . . . . .	3224
6.676.2.11isRemoveSubscriptionInfo . . . . .	3224
6.676.2.12setClientId . . . . .	3225
6.676.2.13setConnectionId . . . . .	3225
6.676.2.14setSubscriptionName . . . . .	3225
6.676.2.15toString . . . . .	3225
6.676.2.16visit . . . . .	3225
6.676.3 Field Documentation . . . . .	3226
6.676.3.1 clientId . . . . .	3226
6.676.3.2 connectionId . . . . .	3226

6.676.3.3 ID_REMOVESUBSCRIPTIONINFO . . . . .	3226
6.676.3.4 subscriptionName . . . . .	3226
6.677activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3227
6.677.1 Detailed Description . . . . .	3227
6.677.2 Constructor & Destructor Documentation . . . . .	3228
6.677.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3228
6.677.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3228
6.677.3 Member Function Documentation . . . . .	3228
6.677.3.1 createObject . . . . .	3228
6.677.3.2 getDataStructureType . . . . .	3228
6.677.3.3 looseMarshal . . . . .	3228
6.677.3.4 looseUnmarshal . . . . .	3229
6.677.3.5 tightMarshal1 . . . . .	3229
6.677.3.6 tightMarshal2 . . . . .	3229
6.677.3.7 tightUnmarshal . . . . .	3230
6.678activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3231
6.678.1 Detailed Description . . . . .	3231
6.678.2 Constructor & Destructor Documentation . . . . .	3232
6.678.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3232
6.678.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3232
6.678.3 Member Function Documentation . . . . .	3232
6.678.3.1 createObject . . . . .	3232
6.678.3.2 getDataStructureType . . . . .	3232
6.678.3.3 looseMarshal . . . . .	3232
6.678.3.4 looseUnmarshal . . . . .	3233
6.678.3.5 tightMarshal1 . . . . .	3233
6.678.3.6 tightMarshal2 . . . . .	3233
6.678.3.7 tightUnmarshal . . . . .	3234
6.679activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3235
6.679.1 Detailed Description . . . . .	3235
6.679.2 Constructor & Destructor Documentation . . . . .	3236
6.679.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3236
6.679.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3236
6.679.3 Member Function Documentation . . . . .	3236

6.679.3.1	createObject . . . . .	3236
6.679.3.2	getDataStructureType . . . . .	3236
6.679.3.3	looseMarshal . . . . .	3236
6.679.3.4	looseUnmarshal . . . . .	3237
6.679.3.5	tightMarshal1 . . . . .	3237
6.679.3.6	tightMarshal2 . . . . .	3237
6.679.3.7	tightUnmarshal . . . . .	3238
6.680	activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	
	Class Reference . . . . .	3239
6.680.1	Detailed Description . . . . .	3239
6.680.2	Constructor & Destructor Documentation . . . . .	3240
	6.680.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3240
	6.680.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3240
6.680.3	Member Function Documentation . . . . .	3240
	6.680.3.1 createObject . . . . .	3240
	6.680.3.2 getDataStructureType . . . . .	3240
	6.680.3.3 looseMarshal . . . . .	3240
	6.680.3.4 looseUnmarshal . . . . .	3241
	6.680.3.5 tightMarshal1 . . . . .	3241
	6.680.3.6 tightMarshal2 . . . . .	3241
	6.680.3.7 tightUnmarshal . . . . .	3242
6.681	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
	Class Reference . . . . .	3243
6.681.1	Detailed Description . . . . .	3243
6.681.2	Constructor & Destructor Documentation . . . . .	3244
	6.681.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3244
	6.681.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3244
6.681.3	Member Function Documentation . . . . .	3244
	6.681.3.1 createObject . . . . .	3244
	6.681.3.2 getDataStructureType . . . . .	3244
	6.681.3.3 looseMarshal . . . . .	3244
	6.681.3.4 looseUnmarshal . . . . .	3245
	6.681.3.5 tightMarshal1 . . . . .	3245
	6.681.3.6 tightMarshal2 . . . . .	3245
	6.681.3.7 tightUnmarshal . . . . .	3246
6.682	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	
	Class Reference . . . . .	3247

6.682.1 Detailed Description . . . . .	3247
6.682.2 Constructor & Destructor Documentation . . . . .	3248
6.682.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3248
6.682.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3248
6.682.3 Member Function Documentation . . . . .	3248
6.682.3.1 createObject . . . . .	3248
6.682.3.2 getDataStructureType . . . . .	3248
6.682.3.3 looseMarshal . . . . .	3248
6.682.3.4 looseUnmarshal . . . . .	3249
6.682.3.5 tightMarshal1 . . . . .	3249
6.682.3.6 tightMarshal2 . . . . .	3249
6.682.3.7 tightUnmarshal . . . . .	3250
6.683activemq::commands::ReplayCommand Class Reference . . . . .	3251
6.683.1 Constructor & Destructor Documentation . . . . .	3252
6.683.1.1 ReplayCommand . . . . .	3252
6.683.1.2 ~ReplayCommand . . . . .	3252
6.683.2 Member Function Documentation . . . . .	3252
6.683.2.1 cloneDataStructure . . . . .	3252
6.683.2.2 copyDataStructure . . . . .	3252
6.683.2.3 equals . . . . .	3252
6.683.2.4 getDataStructureType . . . . .	3252
6.683.2.5 getFirstNakNumber . . . . .	3253
6.683.2.6 getLastNakNumber . . . . .	3253
6.683.2.7 setFirstNakNumber . . . . .	3253
6.683.2.8 setLastNakNumber . . . . .	3253
6.683.2.9 toString . . . . .	3253
6.683.2.10visit . . . . .	3253
6.683.3 Field Documentation . . . . .	3253
6.683.3.1 firstNakNumber . . . . .	3253
6.683.3.2 ID_REPLAYCOMMAND . . . . .	3253
6.683.3.3 lastNakNumber . . . . .	3253
6.684activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference . . . . .	3254
6.684.1 Detailed Description . . . . .	3254
6.684.2 Constructor & Destructor Documentation . . . . .	3255
6.684.2.1 ReplayCommandMarshaller . . . . .	3255
6.684.2.2 ~ReplayCommandMarshaller . . . . .	3255

6.684.3 Member Function Documentation . . . . .	3255
6.684.3.1 createObject . . . . .	3255
6.684.3.2 getDataStructureType . . . . .	3255
6.684.3.3 looseMarshal . . . . .	3255
6.684.3.4 looseUnmarshal . . . . .	3256
6.684.3.5 tightMarshal1 . . . . .	3256
6.684.3.6 tightMarshal2 . . . . .	3256
6.684.3.7 tightUnmarshal . . . . .	3257
6.685activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class	
Reference . . . . .	3258
6.685.1 Detailed Description . . . . .	3258
6.685.2 Constructor & Destructor Documentation . . . . .	3259
6.685.2.1 ReplayCommandMarshaller . . . . .	3259
6.685.2.2 ~ReplayCommandMarshaller . . . . .	3259
6.685.3 Member Function Documentation . . . . .	3259
6.685.3.1 createObject . . . . .	3259
6.685.3.2 getDataStructureType . . . . .	3259
6.685.3.3 looseMarshal . . . . .	3259
6.685.3.4 looseUnmarshal . . . . .	3260
6.685.3.5 tightMarshal1 . . . . .	3260
6.685.3.6 tightMarshal2 . . . . .	3260
6.685.3.7 tightUnmarshal . . . . .	3261
6.686activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class	
Reference . . . . .	3262
6.686.1 Detailed Description . . . . .	3262
6.686.2 Constructor & Destructor Documentation . . . . .	3263
6.686.2.1 ReplayCommandMarshaller . . . . .	3263
6.686.2.2 ~ReplayCommandMarshaller . . . . .	3263
6.686.3 Member Function Documentation . . . . .	3263
6.686.3.1 createObject . . . . .	3263
6.686.3.2 getDataStructureType . . . . .	3263
6.686.3.3 looseMarshal . . . . .	3263
6.686.3.4 looseUnmarshal . . . . .	3264
6.686.3.5 tightMarshal1 . . . . .	3264
6.686.3.6 tightMarshal2 . . . . .	3264
6.686.3.7 tightUnmarshal . . . . .	3265

6.687	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	Class	
	Reference		3266
6.687.1	Detailed Description		3266
6.687.2	Constructor & Destructor Documentation		3267
6.687.2.1	ReplayCommandMarshaller		3267
6.687.2.2	~ReplayCommandMarshaller		3267
6.687.3	Member Function Documentation		3267
6.687.3.1	createObject		3267
6.687.3.2	getDataStructureType		3267
6.687.3.3	looseMarshal		3267
6.687.3.4	looseUnmarshal		3268
6.687.3.5	tightMarshal1		3268
6.687.3.6	tightMarshal2		3268
6.687.3.7	tightUnmarshal		3269
6.688	activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	Class	
	Reference		3270
6.688.1	Detailed Description		3270
6.688.2	Constructor & Destructor Documentation		3271
6.688.2.1	ReplayCommandMarshaller		3271
6.688.2.2	~ReplayCommandMarshaller		3271
6.688.3	Member Function Documentation		3271
6.688.3.1	createObject		3271
6.688.3.2	getDataStructureType		3271
6.688.3.3	looseMarshal		3271
6.688.3.4	looseUnmarshal		3272
6.688.3.5	tightMarshal1		3272
6.688.3.6	tightMarshal2		3272
6.688.3.7	tightUnmarshal		3273
6.689	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	Class	
	Reference		3274
6.689.1	Detailed Description		3274
6.689.2	Constructor & Destructor Documentation		3275
6.689.2.1	ReplayCommandMarshaller		3275
6.689.2.2	~ReplayCommandMarshaller		3275
6.689.3	Member Function Documentation		3275
6.689.3.1	createObject		3275
6.689.3.2	getDataStructureType		3275

6.689.3.3 looseMarshal . . . . .	3275
6.689.3.4 looseUnmarshal . . . . .	3276
6.689.3.5 tightMarshal1 . . . . .	3276
6.689.3.6 tightMarshal2 . . . . .	3276
6.689.3.7 tightUnmarshal . . . . .	3277
6.690activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference . . . .	3278
6.690.1 Constructor & Destructor Documentation . . . . .	3278
6.690.1.1 ResolveProducerExecutor . . . . .	3278
6.690.1.2 ~ResolveProducerExecutor . . . . .	3278
6.690.2 Member Function Documentation . . . . .	3278
6.690.2.1 getDestination . . . . .	3278
6.690.2.2 operator= . . . . .	3278
6.691activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference . . . .	3279
6.691.1 Constructor & Destructor Documentation . . . . .	3279
6.691.1.1 ResolveReceiveExecutor . . . . .	3279
6.691.1.2 ~ResolveReceiveExecutor . . . . .	3279
6.691.2 Member Function Documentation . . . . .	3279
6.691.2.1 getDestination . . . . .	3279
6.691.2.2 operator= . . . . .	3279
6.692decaf::internal::util::Resource Class Reference . . . . .	3280
6.692.1 Detailed Description . . . . .	3280
6.692.2 Constructor & Destructor Documentation . . . . .	3280
6.692.2.1 ~Resource . . . . .	3280
6.693decaf::internal::util::ResourceLifecycleManager Class Reference . . . . .	3281
6.693.1 Detailed Description . . . . .	3281
6.693.2 Constructor & Destructor Documentation . . . . .	3281
6.693.2.1 ResourceLifecycleManager . . . . .	3281
6.693.2.2 ~ResourceLifecycleManager . . . . .	3281
6.693.3 Member Function Documentation . . . . .	3281
6.693.3.1 addResource . . . . .	3281
6.693.3.2 destroyResources . . . . .	3281
6.694activemq::cmsutil::ResourceLifecycleManager Class Reference . . . . .	3282
6.694.1 Detailed Description . . . . .	3282
6.694.2 Constructor & Destructor Documentation . . . . .	3283
6.694.2.1 ResourceLifecycleManager . . . . .	3283
6.694.2.2 ResourceLifecycleManager . . . . .	3283



6.694.2.3 ~ResourceLifecycleManager . . . . .	3283
6.694.3 Member Function Documentation . . . . .	3283
6.694.3.1 addConnection . . . . .	3283
6.694.3.2 addDestination . . . . .	3283
6.694.3.3 addMessageConsumer . . . . .	3283
6.694.3.4 addMessageProducer . . . . .	3283
6.694.3.5 addSession . . . . .	3284
6.694.3.6 destroy . . . . .	3284
6.694.3.7 operator= . . . . .	3284
6.694.3.8 releaseAll . . . . .	3284
6.695activemq::commands::Response Class Reference . . . . .	3285
6.695.1 Constructor & Destructor Documentation . . . . .	3286
6.695.1.1 Response . . . . .	3286
6.695.1.2 ~Response . . . . .	3286
6.695.2 Member Function Documentation . . . . .	3286
6.695.2.1 cloneDataStructure . . . . .	3286
6.695.2.2 copyDataStructure . . . . .	3286
6.695.2.3 equals . . . . .	3286
6.695.2.4 getCorrelationId . . . . .	3287
6.695.2.5 getDataStructureType . . . . .	3287
6.695.2.6 isResponse . . . . .	3287
6.695.2.7 setCorrelationId . . . . .	3287
6.695.2.8 toString . . . . .	3287
6.695.2.9 visit . . . . .	3287
6.695.3 Field Documentation . . . . .	3288
6.695.3.1 correlationId . . . . .	3288
6.695.3.2 ID_RESPONSE . . . . .	3288
6.696activemq::transport::mock::ResponseBuilder Class Reference . . . . .	3289
6.696.1 Detailed Description . . . . .	3289
6.696.2 Constructor & Destructor Documentation . . . . .	3289
6.696.2.1 ~ResponseBuilder . . . . .	3289
6.696.3 Member Function Documentation . . . . .	3289
6.696.3.1 buildIncomingCommands . . . . .	3289
6.696.3.2 buildResponse . . . . .	3290
6.697activemq::transport::correlator::ResponseCorrelator Class Reference . . . . .	3291
6.697.1 Detailed Description . . . . .	3291

6.697.2 Constructor & Destructor Documentation . . . . .	3292
6.697.2.1 ResponseCorrelator . . . . .	3292
6.697.2.2 ~ResponseCorrelator . . . . .	3292
6.697.3 Member Function Documentation . . . . .	3292
6.697.3.1 close . . . . .	3292
6.697.3.2 onCommand . . . . .	3292
6.697.3.3 oneway . . . . .	3292
6.697.3.4 onTransportException . . . . .	3293
6.697.3.5 request . . . . .	3293
6.697.3.6 request . . . . .	3293
6.697.3.7 start . . . . .	3294
6.698activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference . . . . .	3295
6.698.1 Detailed Description . . . . .	3295
6.698.2 Constructor & Destructor Documentation . . . . .	3296
6.698.2.1 ResponseMarshaller . . . . .	3296
6.698.2.2 ~ResponseMarshaller . . . . .	3296
6.698.3 Member Function Documentation . . . . .	3296
6.698.3.1 createObject . . . . .	3296
6.698.3.2 getDataStructureType . . . . .	3296
6.698.3.3 looseMarshal . . . . .	3296
6.698.3.4 looseUnmarshal . . . . .	3297
6.698.3.5 tightMarshal1 . . . . .	3297
6.698.3.6 tightMarshal2 . . . . .	3298
6.698.3.7 tightUnmarshal . . . . .	3299
6.699activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference . . . . .	3300
6.699.1 Detailed Description . . . . .	3300
6.699.2 Constructor & Destructor Documentation . . . . .	3301
6.699.2.1 ResponseMarshaller . . . . .	3301
6.699.2.2 ~ResponseMarshaller . . . . .	3301
6.699.3 Member Function Documentation . . . . .	3301
6.699.3.1 createObject . . . . .	3301
6.699.3.2 getDataStructureType . . . . .	3301
6.699.3.3 looseMarshal . . . . .	3301
6.699.3.4 looseUnmarshal . . . . .	3302
6.699.3.5 tightMarshal1 . . . . .	3302
6.699.3.6 tightMarshal2 . . . . .	3303

6.699.3.7 tightUnmarshal . . . . .	3304
6.700activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference . . . . .	3305
6.700.1 Detailed Description . . . . .	3305
6.700.2 Constructor & Destructor Documentation . . . . .	3306
6.700.2.1 ResponseMarshaller . . . . .	3306
6.700.2.2 ~ResponseMarshaller . . . . .	3306
6.700.3 Member Function Documentation . . . . .	3306
6.700.3.1 createObject . . . . .	3306
6.700.3.2 getDataStructureType . . . . .	3306
6.700.3.3 looseMarshal . . . . .	3306
6.700.3.4 looseUnmarshal . . . . .	3307
6.700.3.5 tightMarshal1 . . . . .	3307
6.700.3.6 tightMarshal2 . . . . .	3308
6.700.3.7 tightUnmarshal . . . . .	3309
6.701activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference . . . . .	3310
6.701.1 Detailed Description . . . . .	3310
6.701.2 Constructor & Destructor Documentation . . . . .	3311
6.701.2.1 ResponseMarshaller . . . . .	3311
6.701.2.2 ~ResponseMarshaller . . . . .	3311
6.701.3 Member Function Documentation . . . . .	3311
6.701.3.1 createObject . . . . .	3311
6.701.3.2 getDataStructureType . . . . .	3311
6.701.3.3 looseMarshal . . . . .	3311
6.701.3.4 looseUnmarshal . . . . .	3312
6.701.3.5 tightMarshal1 . . . . .	3312
6.701.3.6 tightMarshal2 . . . . .	3313
6.701.3.7 tightUnmarshal . . . . .	3314
6.702activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference . . . . .	3315
6.702.1 Detailed Description . . . . .	3315
6.702.2 Constructor & Destructor Documentation . . . . .	3316
6.702.2.1 ResponseMarshaller . . . . .	3316
6.702.2.2 ~ResponseMarshaller . . . . .	3316
6.702.3 Member Function Documentation . . . . .	3316
6.702.3.1 createObject . . . . .	3316
6.702.3.2 getDataStructureType . . . . .	3316
6.702.3.3 looseMarshal . . . . .	3316

6.702.3.4 looseUnmarshal . . . . .	3317
6.702.3.5 tightMarshal1 . . . . .	3317
6.702.3.6 tightMarshal2 . . . . .	3318
6.702.3.7 tightUnmarshal . . . . .	3319
6.703activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference . . . . .	3320
6.703.1 Detailed Description . . . . .	3320
6.703.2 Constructor & Destructor Documentation . . . . .	3321
6.703.2.1 ResponseMarshaller . . . . .	3321
6.703.2.2 ~ResponseMarshaller . . . . .	3321
6.703.3 Member Function Documentation . . . . .	3321
6.703.3.1 createObject . . . . .	3321
6.703.3.2 getDataStructureType . . . . .	3321
6.703.3.3 looseMarshal . . . . .	3321
6.703.3.4 looseUnmarshal . . . . .	3322
6.703.3.5 tightMarshal1 . . . . .	3322
6.703.3.6 tightMarshal2 . . . . .	3323
6.703.3.7 tightUnmarshal . . . . .	3324
6.704decaf::lang::Runnable Class Reference . . . . .	3325
6.704.1 Detailed Description . . . . .	3325
6.704.2 Constructor & Destructor Documentation . . . . .	3325
6.704.2.1 ~Runnable . . . . .	3325
6.704.3 Member Function Documentation . . . . .	3325
6.704.3.1 run . . . . .	3325
6.705decaf::lang::Runtime Class Reference . . . . .	3326
6.705.1 Constructor & Destructor Documentation . . . . .	3326
6.705.1.1 ~Runtime . . . . .	3326
6.705.2 Member Function Documentation . . . . .	3326
6.705.2.1 getRuntime . . . . .	3326
6.705.2.2 initializeRuntime . . . . .	3326
6.705.2.3 initializeRuntime . . . . .	3327
6.705.2.4 shutdownRuntime . . . . .	3327
6.706decaf::lang::exceptions::RuntimeException Class Reference . . . . .	3328
6.706.1 Constructor & Destructor Documentation . . . . .	3328
6.706.1.1 RuntimeException . . . . .	3328
6.706.1.2 RuntimeException . . . . .	3328
6.706.1.3 RuntimeException . . . . .	3329

6.706.1.4 RuntimeException . . . . .	3329
6.706.1.5 RuntimeException . . . . .	3329
6.706.1.6 RuntimeException . . . . .	3329
6.706.1.7 ~RuntimeException . . . . .	3330
6.706.2 Member Function Documentation . . . . .	3330
6.706.2.1 clone . . . . .	3330
6.707decaf::security::SecureRandom Class Reference . . . . .	3331
6.707.1 Detailed Description . . . . .	3332
6.707.2 Constructor & Destructor Documentation . . . . .	3332
6.707.2.1 SecureRandom . . . . .	3332
6.707.2.2 SecureRandom . . . . .	3332
6.707.2.3 SecureRandom . . . . .	3333
6.707.2.4 ~SecureRandom . . . . .	3333
6.707.3 Member Function Documentation . . . . .	3333
6.707.3.1 next . . . . .	3333
6.707.3.2 nextBytes . . . . .	3334
6.707.3.3 nextBytes . . . . .	3334
6.707.3.4 setSeed . . . . .	3334
6.707.3.5 setSeed . . . . .	3335
6.707.3.6 setSeed . . . . .	3335
6.708decaf::internal::security::SecureRandomImpl Class Reference . . . . .	3336
6.708.1 Detailed Description . . . . .	3336
6.708.2 Constructor & Destructor Documentation . . . . .	3337
6.708.2.1 SecureRandomImpl . . . . .	3337
6.708.2.2 ~SecureRandomImpl . . . . .	3337
6.708.2.3 SecureRandomImpl . . . . .	3337
6.708.2.4 ~SecureRandomImpl . . . . .	3337
6.708.3 Member Function Documentation . . . . .	3337
6.708.3.1 providerGenerateSeed . . . . .	3337
6.708.3.2 providerGenerateSeed . . . . .	3337
6.708.3.3 providerNextBytes . . . . .	3337
6.708.3.4 providerNextBytes . . . . .	3338
6.708.3.5 providerSetSeed . . . . .	3338
6.708.3.6 providerSetSeed . . . . .	3338
6.709decaf::security::SecureRandomSpi Class Reference . . . . .	3339
6.709.1 Detailed Description . . . . .	3339

6.709.2 Constructor & Destructor Documentation . . . . .	3339
6.709.2.1 SecureRandomSpi . . . . .	3339
6.709.2.2 ~SecureRandomSpi . . . . .	3339
6.709.3 Member Function Documentation . . . . .	3339
6.709.3.1 providerGenerateSeed . . . . .	3339
6.709.3.2 providerNextBytes . . . . .	3340
6.709.3.3 providerSetSeed . . . . .	3340
6.710decaf::util::concurrent::Semaphore Class Reference . . . . .	3341
6.710.1 Detailed Description . . . . .	3342
6.710.2 Constructor & Destructor Documentation . . . . .	3343
6.710.2.1 Semaphore . . . . .	3343
6.710.2.2 Semaphore . . . . .	3343
6.710.2.3 ~Semaphore . . . . .	3344
6.710.3 Member Function Documentation . . . . .	3344
6.710.3.1 acquire . . . . .	3344
6.710.3.2 acquire . . . . .	3344
6.710.3.3 acquireUninterruptibly . . . . .	3345
6.710.3.4 acquireUninterruptibly . . . . .	3345
6.710.3.5 availablePermits . . . . .	3346
6.710.3.6 drainPermits . . . . .	3346
6.710.3.7 isFair . . . . .	3346
6.710.3.8 release . . . . .	3346
6.710.3.9 release . . . . .	3347
6.710.3.10 toString . . . . .	3347
6.710.3.11 tryAcquire . . . . .	3347
6.710.3.12 tryAcquire . . . . .	3348
6.710.3.13 tryAcquire . . . . .	3348
6.710.3.14 tryAcquire . . . . .	3349
6.711activemq::cmsutil::CmsTemplate::SendExecutor Class Reference . . . . .	3350
6.711.1 Constructor & Destructor Documentation . . . . .	3350
6.711.1.1 SendExecutor . . . . .	3350
6.711.1.2 SendExecutor . . . . .	3350
6.711.1.3 ~SendExecutor . . . . .	3350
6.711.2 Member Function Documentation . . . . .	3350
6.711.2.1 doInCms . . . . .	3350
6.711.2.2 operator= . . . . .	3351

6.712	decaf::net::ServerSocket Class Reference . . . . .	3352
6.712.1	Detailed Description . . . . .	3354
6.712.2	Constructor & Destructor Documentation . . . . .	3354
6.712.2.1	ServerSocket . . . . .	3354
6.712.2.2	ServerSocket . . . . .	3354
6.712.2.3	ServerSocket . . . . .	3354
6.712.2.4	ServerSocket . . . . .	3355
6.712.2.5	~ServerSocket . . . . .	3355
6.712.2.6	ServerSocket . . . . .	3355
6.712.3	Member Function Documentation . . . . .	3356
6.712.3.1	accept . . . . .	3356
6.712.3.2	bind . . . . .	3356
6.712.3.3	bind . . . . .	3356
6.712.3.4	checkClosed . . . . .	3357
6.712.3.5	close . . . . .	3357
6.712.3.6	ensureCreated . . . . .	3357
6.712.3.7	getDefaultBacklog . . . . .	3357
6.712.3.8	getLocalPort . . . . .	3357
6.712.3.9	getReceiveBufferSize . . . . .	3357
6.712.3.10	getReuseAddress . . . . .	3358
6.712.3.11	getSoTimeout . . . . .	3358
6.712.3.12	implAccept . . . . .	3358
6.712.3.13	sBound . . . . .	3358
6.712.3.14	sClosed . . . . .	3358
6.712.3.15	setReceiveBufferSize . . . . .	3359
6.712.3.16	setReuseAddress . . . . .	3359
6.712.3.17	setSocketImplFactory . . . . .	3359
6.712.3.18	setSoTimeout . . . . .	3359
6.712.3.19	setUpSocketImpl . . . . .	3360
6.712.3.20	toString . . . . .	3360
6.713	decaf::net::ServerSocketFactory Class Reference . . . . .	3361
6.713.1	Detailed Description . . . . .	3361
6.713.2	Constructor & Destructor Documentation . . . . .	3362
6.713.2.1	ServerSocketFactory . . . . .	3362
6.713.2.2	~ServerSocketFactory . . . . .	3362
6.713.3	Member Function Documentation . . . . .	3362

6.713.3.1 createServerSocket . . . . .	3362
6.713.3.2 createServerSocket . . . . .	3362
6.713.3.3 createServerSocket . . . . .	3363
6.713.3.4 createServerSocket . . . . .	3363
6.713.3.5 getDefault . . . . .	3363
6.714cms::Session Class Reference . . . . .	3365
6.714.1 Detailed Description . . . . .	3367
6.714.2 Member Enumeration Documentation . . . . .	3368
6.714.2.1 AcknowledgeMode . . . . .	3368
6.714.3 Constructor & Destructor Documentation . . . . .	3368
6.714.3.1 ~Session . . . . .	3368
6.714.4 Member Function Documentation . . . . .	3368
6.714.4.1 close . . . . .	3368
6.714.4.2 commit . . . . .	3369
6.714.4.3 createBrowser . . . . .	3369
6.714.4.4 createBrowser . . . . .	3369
6.714.4.5 createBytesMessage . . . . .	3370
6.714.4.6 createBytesMessage . . . . .	3370
6.714.4.7 createConsumer . . . . .	3370
6.714.4.8 createConsumer . . . . .	3371
6.714.4.9 createConsumer . . . . .	3371
6.714.4.10 createDurableConsumer . . . . .	3372
6.714.4.11 createMapMessage . . . . .	3372
6.714.4.12 createMessage . . . . .	3372
6.714.4.13 createProducer . . . . .	3373
6.714.4.14 createQueue . . . . .	3373
6.714.4.15 createStreamMessage . . . . .	3373
6.714.4.16 createTemporaryQueue . . . . .	3374
6.714.4.17 createTemporaryTopic . . . . .	3374
6.714.4.18 createTextMessage . . . . .	3374
6.714.4.19 createTextMessage . . . . .	3374
6.714.4.20 createTopic . . . . .	3375
6.714.4.21 getAcknowledgeMode . . . . .	3375
6.714.4.22 isTransacted . . . . .	3375
6.714.4.23 recover . . . . .	3376
6.714.4.24 rollback . . . . .	3376



6.714.4.25	unsubscribe . . . . .	3376
6.715	activemq::cmsutil::SessionCallback Class Reference . . . . .	3378
6.715.1	Detailed Description . . . . .	3378
6.715.2	Constructor & Destructor Documentation . . . . .	3378
6.715.2.1	~SessionCallback . . . . .	3378
6.715.3	Member Function Documentation . . . . .	3378
6.715.3.1	doInCms . . . . .	3378
6.716	activemq::commands::SessionId Class Reference . . . . .	3379
6.716.1	Member Typedef Documentation . . . . .	3380
6.716.1.1	COMPARATOR . . . . .	3380
6.716.2	Constructor & Destructor Documentation . . . . .	3380
6.716.2.1	SessionId . . . . .	3380
6.716.2.2	SessionId . . . . .	3380
6.716.2.3	SessionId . . . . .	3380
6.716.2.4	SessionId . . . . .	3380
6.716.2.5	SessionId . . . . .	3380
6.716.2.6	~SessionId . . . . .	3380
6.716.3	Member Function Documentation . . . . .	3380
6.716.3.1	cloneDataStructure . . . . .	3380
6.716.3.2	compareTo . . . . .	3380
6.716.3.3	copyDataStructure . . . . .	3380
6.716.3.4	equals . . . . .	3381
6.716.3.5	equals . . . . .	3381
6.716.3.6	getConnectionId . . . . .	3381
6.716.3.7	getConnectionId . . . . .	3381
6.716.3.8	getDataStructureType . . . . .	3381
6.716.3.9	getParentId . . . . .	3382
6.716.3.10	getValue . . . . .	3382
6.716.3.11	operator< . . . . .	3382
6.716.3.12	operator= . . . . .	3382
6.716.3.13	operator== . . . . .	3382
6.716.3.14	setConnectionId . . . . .	3382
6.716.3.15	setValue . . . . .	3382
6.716.3.16	toString . . . . .	3382
6.716.4	Field Documentation . . . . .	3382
6.716.4.1	connectionId . . . . .	3382

6.716.4.2 ID_SESSIONID . . . . .	3382
6.716.4.3 value . . . . .	3382
6.717activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference . . . . .	3383
6.717.1 Detailed Description . . . . .	3383
6.717.2 Constructor & Destructor Documentation . . . . .	3384
6.717.2.1 SessionIdMarshaller . . . . .	3384
6.717.2.2 ~SessionIdMarshaller . . . . .	3384
6.717.3 Member Function Documentation . . . . .	3384
6.717.3.1 createObject . . . . .	3384
6.717.3.2 getDataStructureType . . . . .	3384
6.717.3.3 looseMarshal . . . . .	3384
6.717.3.4 looseUnmarshal . . . . .	3385
6.717.3.5 tightMarshal1 . . . . .	3385
6.717.3.6 tightMarshal2 . . . . .	3385
6.717.3.7 tightUnmarshal . . . . .	3386
6.718activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference . . . . .	3387
6.718.1 Detailed Description . . . . .	3387
6.718.2 Constructor & Destructor Documentation . . . . .	3388
6.718.2.1 SessionIdMarshaller . . . . .	3388
6.718.2.2 ~SessionIdMarshaller . . . . .	3388
6.718.3 Member Function Documentation . . . . .	3388
6.718.3.1 createObject . . . . .	3388
6.718.3.2 getDataStructureType . . . . .	3388
6.718.3.3 looseMarshal . . . . .	3388
6.718.3.4 looseUnmarshal . . . . .	3389
6.718.3.5 tightMarshal1 . . . . .	3389
6.718.3.6 tightMarshal2 . . . . .	3389
6.718.3.7 tightUnmarshal . . . . .	3390
6.719activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference . . . . .	3391
6.719.1 Detailed Description . . . . .	3391
6.719.2 Constructor & Destructor Documentation . . . . .	3392
6.719.2.1 SessionIdMarshaller . . . . .	3392
6.719.2.2 ~SessionIdMarshaller . . . . .	3392
6.719.3 Member Function Documentation . . . . .	3392
6.719.3.1 createObject . . . . .	3392
6.719.3.2 getDataStructureType . . . . .	3392

6.719.3.3 looseMarshal . . . . .	3392
6.719.3.4 looseUnmarshal . . . . .	3393
6.719.3.5 tightMarshal1 . . . . .	3393
6.719.3.6 tightMarshal2 . . . . .	3393
6.719.3.7 tightUnmarshal . . . . .	3394
6.720activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference . . . . .	3395
6.720.1 Detailed Description . . . . .	3395
6.720.2 Constructor & Destructor Documentation . . . . .	3396
6.720.2.1 SessionIdMarshaller . . . . .	3396
6.720.2.2 ~SessionIdMarshaller . . . . .	3396
6.720.3 Member Function Documentation . . . . .	3396
6.720.3.1 createObject . . . . .	3396
6.720.3.2 getDataStructureType . . . . .	3396
6.720.3.3 looseMarshal . . . . .	3396
6.720.3.4 looseUnmarshal . . . . .	3397
6.720.3.5 tightMarshal1 . . . . .	3397
6.720.3.6 tightMarshal2 . . . . .	3397
6.720.3.7 tightUnmarshal . . . . .	3398
6.721activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference . . . . .	3399
6.721.1 Detailed Description . . . . .	3399
6.721.2 Constructor & Destructor Documentation . . . . .	3400
6.721.2.1 SessionIdMarshaller . . . . .	3400
6.721.2.2 ~SessionIdMarshaller . . . . .	3400
6.721.3 Member Function Documentation . . . . .	3400
6.721.3.1 createObject . . . . .	3400
6.721.3.2 getDataStructureType . . . . .	3400
6.721.3.3 looseMarshal . . . . .	3400
6.721.3.4 looseUnmarshal . . . . .	3401
6.721.3.5 tightMarshal1 . . . . .	3401
6.721.3.6 tightMarshal2 . . . . .	3401
6.721.3.7 tightUnmarshal . . . . .	3402
6.722activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference . . . . .	3403
6.722.1 Detailed Description . . . . .	3403
6.722.2 Constructor & Destructor Documentation . . . . .	3404
6.722.2.1 SessionIdMarshaller . . . . .	3404
6.722.2.2 ~SessionIdMarshaller . . . . .	3404

6.722.3 Member Function Documentation . . . . .	3404
6.722.3.1 createObject . . . . .	3404
6.722.3.2 getDataStructureType . . . . .	3404
6.722.3.3 looseMarshal . . . . .	3404
6.722.3.4 looseUnmarshal . . . . .	3405
6.722.3.5 tightMarshal1 . . . . .	3405
6.722.3.6 tightMarshal2 . . . . .	3405
6.722.3.7 tightUnmarshal . . . . .	3406
6.723activemq::commands::SessionInfo Class Reference . . . . .	3407
6.723.1 Constructor & Destructor Documentation . . . . .	3408
6.723.1.1 SessionInfo . . . . .	3408
6.723.1.2 ~SessionInfo . . . . .	3408
6.723.2 Member Function Documentation . . . . .	3408
6.723.2.1 cloneDataStructure . . . . .	3408
6.723.2.2 copyDataStructure . . . . .	3408
6.723.2.3 createRemoveCommand . . . . .	3408
6.723.2.4 equals . . . . .	3408
6.723.2.5 getAckMode . . . . .	3409
6.723.2.6 getDataStructureType . . . . .	3409
6.723.2.7 getSessionId . . . . .	3409
6.723.2.8 getSessionId . . . . .	3409
6.723.2.9 setAckMode . . . . .	3409
6.723.2.10setSessionId . . . . .	3409
6.723.2.11toString . . . . .	3409
6.723.2.12visit . . . . .	3409
6.723.3 Field Documentation . . . . .	3410
6.723.3.1 ID_SESSIONINFO . . . . .	3410
6.723.3.2 sessionId . . . . .	3410
6.724activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference	3411
6.724.1 Detailed Description . . . . .	3411
6.724.2 Constructor & Destructor Documentation . . . . .	3412
6.724.2.1 SessionInfoMarshaller . . . . .	3412
6.724.2.2 ~SessionInfoMarshaller . . . . .	3412
6.724.3 Member Function Documentation . . . . .	3412
6.724.3.1 createObject . . . . .	3412
6.724.3.2 getDataStructureType . . . . .	3412

6.724.3.3 looseMarshal . . . . .	3412
6.724.3.4 looseUnmarshal . . . . .	3413
6.724.3.5 tightMarshal1 . . . . .	3413
6.724.3.6 tightMarshal2 . . . . .	3413
6.724.3.7 tightUnmarshal . . . . .	3414
6.725activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference	3415
6.725.1 Detailed Description . . . . .	3415
6.725.2 Constructor & Destructor Documentation . . . . .	3416
6.725.2.1 SessionInfoMarshaller . . . . .	3416
6.725.2.2 ~SessionInfoMarshaller . . . . .	3416
6.725.3 Member Function Documentation . . . . .	3416
6.725.3.1 createObject . . . . .	3416
6.725.3.2 getDataStructureType . . . . .	3416
6.725.3.3 looseMarshal . . . . .	3416
6.725.3.4 looseUnmarshal . . . . .	3417
6.725.3.5 tightMarshal1 . . . . .	3417
6.725.3.6 tightMarshal2 . . . . .	3417
6.725.3.7 tightUnmarshal . . . . .	3418
6.726activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference	3419
6.726.1 Detailed Description . . . . .	3419
6.726.2 Constructor & Destructor Documentation . . . . .	3420
6.726.2.1 SessionInfoMarshaller . . . . .	3420
6.726.2.2 ~SessionInfoMarshaller . . . . .	3420
6.726.3 Member Function Documentation . . . . .	3420
6.726.3.1 createObject . . . . .	3420
6.726.3.2 getDataStructureType . . . . .	3420
6.726.3.3 looseMarshal . . . . .	3420
6.726.3.4 looseUnmarshal . . . . .	3421
6.726.3.5 tightMarshal1 . . . . .	3421
6.726.3.6 tightMarshal2 . . . . .	3421
6.726.3.7 tightUnmarshal . . . . .	3422
6.727activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference	3423
6.727.1 Detailed Description . . . . .	3423
6.727.2 Constructor & Destructor Documentation . . . . .	3424
6.727.2.1 SessionInfoMarshaller . . . . .	3424
6.727.2.2 ~SessionInfoMarshaller . . . . .	3424

6.727.3 Member Function Documentation . . . . .	3424
6.727.3.1 createObject . . . . .	3424
6.727.3.2 getDataStructureType . . . . .	3424
6.727.3.3 looseMarshal . . . . .	3424
6.727.3.4 looseUnmarshal . . . . .	3425
6.727.3.5 tightMarshal1 . . . . .	3425
6.727.3.6 tightMarshal2 . . . . .	3425
6.727.3.7 tightUnmarshal . . . . .	3426
6.728activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference	3427
6.728.1 Detailed Description . . . . .	3427
6.728.2 Constructor & Destructor Documentation . . . . .	3428
6.728.2.1 SessionInfoMarshaller . . . . .	3428
6.728.2.2 ~SessionInfoMarshaller . . . . .	3428
6.728.3 Member Function Documentation . . . . .	3428
6.728.3.1 createObject . . . . .	3428
6.728.3.2 getDataStructureType . . . . .	3428
6.728.3.3 looseMarshal . . . . .	3428
6.728.3.4 looseUnmarshal . . . . .	3429
6.728.3.5 tightMarshal1 . . . . .	3429
6.728.3.6 tightMarshal2 . . . . .	3429
6.728.3.7 tightUnmarshal . . . . .	3430
6.729activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference	3431
6.729.1 Detailed Description . . . . .	3431
6.729.2 Constructor & Destructor Documentation . . . . .	3432
6.729.2.1 SessionInfoMarshaller . . . . .	3432
6.729.2.2 ~SessionInfoMarshaller . . . . .	3432
6.729.3 Member Function Documentation . . . . .	3432
6.729.3.1 createObject . . . . .	3432
6.729.3.2 getDataStructureType . . . . .	3432
6.729.3.3 looseMarshal . . . . .	3432
6.729.3.4 looseUnmarshal . . . . .	3433
6.729.3.5 tightMarshal1 . . . . .	3433
6.729.3.6 tightMarshal2 . . . . .	3433
6.729.3.7 tightUnmarshal . . . . .	3434
6.730activemq::cmsutil::SessionPool Class Reference . . . . .	3435
6.730.1 Detailed Description . . . . .	3435

6.730.2 Constructor & Destructor Documentation . . . . .	3435
6.730.2.1 SessionPool . . . . .	3435
6.730.2.2 SessionPool . . . . .	3435
6.730.2.3 ~SessionPool . . . . .	3436
6.730.3 Member Function Documentation . . . . .	3436
6.730.3.1 getResourceLifecycleManager . . . . .	3436
6.730.3.2 operator= . . . . .	3436
6.730.3.3 returnSession . . . . .	3436
6.730.3.4 takeSession . . . . .	3436
6.731activemq::state::SessionState Class Reference . . . . .	3437
6.731.1 Constructor & Destructor Documentation . . . . .	3438
6.731.1.1 SessionState . . . . .	3438
6.731.1.2 ~SessionState . . . . .	3438
6.731.2 Member Function Documentation . . . . .	3438
6.731.2.1 addConsumer . . . . .	3438
6.731.2.2 addProducer . . . . .	3438
6.731.2.3 checkShutdown . . . . .	3438
6.731.2.4 getConsumerState . . . . .	3438
6.731.2.5 getConsumerStates . . . . .	3438
6.731.2.6 getInfo . . . . .	3438
6.731.2.7 getProducerState . . . . .	3438
6.731.2.8 getProducerStates . . . . .	3438
6.731.2.9 removeConsumer . . . . .	3438
6.731.2.10removeProducer . . . . .	3438
6.731.2.11shutdown . . . . .	3438
6.731.2.12toString . . . . .	3438
6.732decaf::util::Set< E > Class Template Reference . . . . .	3439
6.732.1 Detailed Description . . . . .	3439
6.732.2 Constructor & Destructor Documentation . . . . .	3439
6.732.2.1 ~Set . . . . .	3439
6.733decaf::lang::Short Class Reference . . . . .	3440
6.733.1 Constructor & Destructor Documentation . . . . .	3441
6.733.1.1 Short . . . . .	3441
6.733.1.2 Short . . . . .	3442
6.733.1.3 ~Short . . . . .	3442
6.733.2 Member Function Documentation . . . . .	3442

6.733.2.1	byteValue . . . . .	3442
6.733.2.2	compareTo . . . . .	3442
6.733.2.3	compareTo . . . . .	3442
6.733.2.4	decode . . . . .	3443
6.733.2.5	doubleValue . . . . .	3443
6.733.2.6	equals . . . . .	3443
6.733.2.7	equals . . . . .	3443
6.733.2.8	float Value . . . . .	3444
6.733.2.9	int Value . . . . .	3444
6.733.2.10	longValue . . . . .	3444
6.733.2.11	operator< . . . . .	3444
6.733.2.12	operator< . . . . .	3444
6.733.2.13	operator== . . . . .	3445
6.733.2.14	operator== . . . . .	3445
6.733.2.15	parseShort . . . . .	3445
6.733.2.16	parseShort . . . . .	3446
6.733.2.17	reverseBytes . . . . .	3446
6.733.2.18	short Value . . . . .	3446
6.733.2.19	toString . . . . .	3446
6.733.2.20	toString . . . . .	3447
6.733.2.21	valueOf . . . . .	3447
6.733.2.22	valueOf . . . . .	3447
6.733.2.23	valueOf . . . . .	3447
6.733.3	Field Documentation . . . . .	3448
6.733.3.1	MAX_VALUE . . . . .	3448
6.733.3.2	MIN_VALUE . . . . .	3448
6.733.3.3	SIZE . . . . .	3448
6.734	decaf::internal::nio::ShortArrayBuffer Class Reference . . . . .	3449
6.734.1	Constructor & Destructor Documentation . . . . .	3452
6.734.1.1	ShortArrayBuffer . . . . .	3452
6.734.1.2	ShortArrayBuffer . . . . .	3452
6.734.1.3	ShortArrayBuffer . . . . .	3453
6.734.1.4	ShortArrayBuffer . . . . .	3453
6.734.1.5	~ShortArrayBuffer . . . . .	3453
6.734.2	Member Function Documentation . . . . .	3453
6.734.2.1	array . . . . .	3453



6.734.2.2 arrayOffset . . . . .	3454
6.734.2.3 asReadOnlyBuffer . . . . .	3454
6.734.2.4 compact . . . . .	3455
6.734.2.5 duplicate . . . . .	3455
6.734.2.6 get . . . . .	3455
6.734.2.7 get . . . . .	3456
6.734.2.8 hasArray . . . . .	3456
6.734.2.9 isReadOnly . . . . .	3456
6.734.2.10put . . . . .	3456
6.734.2.11put . . . . .	3457
6.734.2.12setReadOnly . . . . .	3457
6.734.2.13slice . . . . .	3457
6.735decaf::nio::ShortBuffer Class Reference . . . . .	3459
6.735.1 Detailed Description . . . . .	3461
6.735.2 Constructor & Destructor Documentation . . . . .	3461
6.735.2.1 ShortBuffer . . . . .	3461
6.735.2.2 ~ShortBuffer . . . . .	3461
6.735.3 Member Function Documentation . . . . .	3461
6.735.3.1 allocate . . . . .	3461
6.735.3.2 array . . . . .	3462
6.735.3.3 arrayOffset . . . . .	3462
6.735.3.4 asReadOnlyBuffer . . . . .	3462
6.735.3.5 compact . . . . .	3463
6.735.3.6 compareTo . . . . .	3463
6.735.3.7 duplicate . . . . .	3463
6.735.3.8 equals . . . . .	3463
6.735.3.9 get . . . . .	3463
6.735.3.10get . . . . .	3464
6.735.3.11get . . . . .	3464
6.735.3.12get . . . . .	3465
6.735.3.13hasArray . . . . .	3465
6.735.3.14operator< . . . . .	3465
6.735.3.15operator== . . . . .	3465
6.735.3.16put . . . . .	3465
6.735.3.17put . . . . .	3466
6.735.3.18put . . . . .	3466

6.735.3.1	put . . . . .	3466
6.735.3.2	put . . . . .	3467
6.735.3.2	slice . . . . .	3468
6.735.3.2	toString . . . . .	3468
6.735.3.2	wrap . . . . .	3468
6.735.3.2	wrap . . . . .	3468
6.736	activemq::commands::ShutdownInfo Class Reference . . . . .	3470
6.736.1	Constructor & Destructor Documentation . . . . .	3471
6.736.1.1	ShutdownInfo . . . . .	3471
6.736.1.2	~ShutdownInfo . . . . .	3471
6.736.2	Member Function Documentation . . . . .	3471
6.736.2.1	cloneDataStructure . . . . .	3471
6.736.2.2	copyDataStructure . . . . .	3471
6.736.2.3	equals . . . . .	3471
6.736.2.4	getDataStructureType . . . . .	3471
6.736.2.5	isShutdownInfo . . . . .	3472
6.736.2.6	toString . . . . .	3472
6.736.2.7	visit . . . . .	3472
6.736.3	Field Documentation . . . . .	3472
6.736.3.1	ID_SHUTDOWNINFO . . . . .	3472
6.737	activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Refer- ence . . . . .	3473
6.737.1	Detailed Description . . . . .	3473
6.737.2	Constructor & Destructor Documentation . . . . .	3474
6.737.2.1	ShutdownInfoMarshaller . . . . .	3474
6.737.2.2	~ShutdownInfoMarshaller . . . . .	3474
6.737.3	Member Function Documentation . . . . .	3474
6.737.3.1	createObject . . . . .	3474
6.737.3.2	getDataStructureType . . . . .	3474
6.737.3.3	looseMarshal . . . . .	3474
6.737.3.4	looseUnmarshal . . . . .	3475
6.737.3.5	tightMarshal1 . . . . .	3475
6.737.3.6	tightMarshal2 . . . . .	3475
6.737.3.7	tightUnmarshal . . . . .	3476
6.738	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Refer- ence . . . . .	3477
6.738.1	Detailed Description . . . . .	3477

6.738.2 Constructor & Destructor Documentation . . . . .	3478
6.738.2.1 ShutdownInfoMarshaller . . . . .	3478
6.738.2.2 ~ShutdownInfoMarshaller . . . . .	3478
6.738.3 Member Function Documentation . . . . .	3478
6.738.3.1 createObject . . . . .	3478
6.738.3.2 getDataStructureType . . . . .	3478
6.738.3.3 looseMarshal . . . . .	3478
6.738.3.4 looseUnmarshal . . . . .	3479
6.738.3.5 tightMarshal1 . . . . .	3479
6.738.3.6 tightMarshal2 . . . . .	3479
6.738.3.7 tightUnmarshal . . . . .	3480
6.739activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Refer- ence . . . . .	3481
6.739.1 Detailed Description . . . . .	3481
6.739.2 Constructor & Destructor Documentation . . . . .	3482
6.739.2.1 ShutdownInfoMarshaller . . . . .	3482
6.739.2.2 ~ShutdownInfoMarshaller . . . . .	3482
6.739.3 Member Function Documentation . . . . .	3482
6.739.3.1 createObject . . . . .	3482
6.739.3.2 getDataStructureType . . . . .	3482
6.739.3.3 looseMarshal . . . . .	3482
6.739.3.4 looseUnmarshal . . . . .	3483
6.739.3.5 tightMarshal1 . . . . .	3483
6.739.3.6 tightMarshal2 . . . . .	3483
6.739.3.7 tightUnmarshal . . . . .	3484
6.740activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Refer- ence . . . . .	3485
6.740.1 Detailed Description . . . . .	3485
6.740.2 Constructor & Destructor Documentation . . . . .	3486
6.740.2.1 ShutdownInfoMarshaller . . . . .	3486
6.740.2.2 ~ShutdownInfoMarshaller . . . . .	3486
6.740.3 Member Function Documentation . . . . .	3486
6.740.3.1 createObject . . . . .	3486
6.740.3.2 getDataStructureType . . . . .	3486
6.740.3.3 looseMarshal . . . . .	3486
6.740.3.4 looseUnmarshal . . . . .	3487
6.740.3.5 tightMarshal1 . . . . .	3487

6.740.3.6 tightMarshal2 . . . . .	3487
6.740.3.7 tightUnmarshal . . . . .	3488
6.741activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference . . . . .	3489
6.741.1 Detailed Description . . . . .	3489
6.741.2 Constructor & Destructor Documentation . . . . .	3490
6.741.2.1 ShutdownInfoMarshaller . . . . .	3490
6.741.2.2 ~ShutdownInfoMarshaller . . . . .	3490
6.741.3 Member Function Documentation . . . . .	3490
6.741.3.1 createObject . . . . .	3490
6.741.3.2 getDataStructureType . . . . .	3490
6.741.3.3 looseMarshal . . . . .	3490
6.741.3.4 looseUnmarshal . . . . .	3491
6.741.3.5 tightMarshal1 . . . . .	3491
6.741.3.6 tightMarshal2 . . . . .	3491
6.741.3.7 tightUnmarshal . . . . .	3492
6.742activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference . . . . .	3493
6.742.1 Detailed Description . . . . .	3493
6.742.2 Constructor & Destructor Documentation . . . . .	3494
6.742.2.1 ShutdownInfoMarshaller . . . . .	3494
6.742.2.2 ~ShutdownInfoMarshaller . . . . .	3494
6.742.3 Member Function Documentation . . . . .	3494
6.742.3.1 createObject . . . . .	3494
6.742.3.2 getDataStructureType . . . . .	3494
6.742.3.3 looseMarshal . . . . .	3494
6.742.3.4 looseUnmarshal . . . . .	3495
6.742.3.5 tightMarshal1 . . . . .	3495
6.742.3.6 tightMarshal2 . . . . .	3495
6.742.3.7 tightUnmarshal . . . . .	3496
6.743decaf::security::SignatureException Class Reference . . . . .	3497
6.743.1 Constructor & Destructor Documentation . . . . .	3497
6.743.1.1 SignatureException . . . . .	3497
6.743.1.2 SignatureException . . . . .	3497
6.743.1.3 SignatureException . . . . .	3498
6.743.1.4 SignatureException . . . . .	3498
6.743.1.5 SignatureException . . . . .	3498

6.743.1.6 SignatureException . . . . .	3498
6.743.1.7 ~SignatureException . . . . .	3499
6.743.2 Member Function Documentation . . . . .	3499
6.743.2.1 clone . . . . .	3499
6.744decaf::util::logging::SimpleFormatter Class Reference . . . . .	3500
6.744.1 Detailed Description . . . . .	3500
6.744.2 Constructor & Destructor Documentation . . . . .	3500
6.744.2.1 SimpleFormatter . . . . .	3500
6.744.2.2 ~SimpleFormatter . . . . .	3500
6.744.3 Member Function Documentation . . . . .	3500
6.744.3.1 format . . . . .	3500
6.745decaf::util::logging::SimpleLogger Class Reference . . . . .	3501
6.745.1 Constructor & Destructor Documentation . . . . .	3501
6.745.1.1 SimpleLogger . . . . .	3501
6.745.1.2 ~SimpleLogger . . . . .	3501
6.745.2 Member Function Documentation . . . . .	3502
6.745.2.1 debug . . . . .	3502
6.745.2.2 error . . . . .	3502
6.745.2.3 fatal . . . . .	3502
6.745.2.4 info . . . . .	3502
6.745.2.5 log . . . . .	3502
6.745.2.6 mark . . . . .	3502
6.745.2.7 warn . . . . .	3502
6.746decaf::net::Socket Class Reference . . . . .	3503
6.746.1 Detailed Description . . . . .	3506
6.746.2 Constructor & Destructor Documentation . . . . .	3506
6.746.2.1 Socket . . . . .	3506
6.746.2.2 Socket . . . . .	3506
6.746.2.3 Socket . . . . .	3507
6.746.2.4 Socket . . . . .	3507
6.746.2.5 Socket . . . . .	3507
6.746.2.6 Socket . . . . .	3508
6.746.2.7 ~Socket . . . . .	3508
6.746.3 Member Function Documentation . . . . .	3508
6.746.3.1 accepted . . . . .	3508
6.746.3.2 bind . . . . .	3508

6.746.3.3	checkClosed	3509
6.746.3.4	close	3509
6.746.3.5	connect	3509
6.746.3.6	connect	3509
6.746.3.7	ensureCreated	3510
6.746.3.8	getInetAddress	3510
6.746.3.9	getInputStream	3510
6.746.3.10	getKeepAlive	3510
6.746.3.11	getLocalAddress	3511
6.746.3.12	getLocalPort	3511
6.746.3.13	getOOBInline	3511
6.746.3.14	getOutputStream	3511
6.746.3.15	getPort	3511
6.746.3.16	getReceiveBufferSize	3512
6.746.3.17	getReuseAddress	3512
6.746.3.18	getSendBufferSize	3512
6.746.3.19	getSoLinger	3512
6.746.3.20	getSoTimeout	3513
6.746.3.21	getTcpNoDelay	3513
6.746.3.22	getTrafficClass	3513
6.746.3.23	initSocketImpl	3513
6.746.3.24	isBound	3513
6.746.3.25	isClosed	3514
6.746.3.26	isConnected	3514
6.746.3.27	isInputShutdown	3514
6.746.3.28	isOutputShutdown	3514
6.746.3.29	isSendUrgentData	3514
6.746.3.30	setKeepAlive	3514
6.746.3.31	setOOBInline	3515
6.746.3.32	setReceiveBufferSize	3515
6.746.3.33	setReuseAddress	3515
6.746.3.34	setSendBufferSize	3515
6.746.3.35	setSocketImplFactory	3516
6.746.3.36	setSoLinger	3516
6.746.3.37	setSoTimeout	3516
6.746.3.38	setTcpNoDelay	3517

6.746.3.39	setTrafficClass . . . . .	3517
6.746.3.40	shutdownInput . . . . .	3517
6.746.3.41	shutdownOutput . . . . .	3518
6.746.3.42	toString . . . . .	3518
6.746.4	Friends And Related Function Documentation . . . . .	3518
6.746.4.1	ServerSocket . . . . .	3518
6.746.5	Field Documentation . . . . .	3518
6.746.5.1	impl . . . . .	3518
6.747	decaf::net::SocketAddress Class Reference . . . . .	3519
6.747.1	Detailed Description . . . . .	3519
6.747.2	Constructor & Destructor Documentation . . . . .	3519
6.747.2.1	~SocketAddress . . . . .	3519
6.748	decaf::net::SocketError Class Reference . . . . .	3520
6.748.1	Detailed Description . . . . .	3520
6.748.2	Member Function Documentation . . . . .	3520
6.748.2.1	getErrorCode . . . . .	3520
6.748.2.2	getErrorString . . . . .	3520
6.749	decaf::net::SocketException Class Reference . . . . .	3521
6.749.1	Detailed Description . . . . .	3521
6.749.2	Constructor & Destructor Documentation . . . . .	3521
6.749.2.1	SocketException . . . . .	3521
6.749.2.2	SocketException . . . . .	3521
6.749.2.3	SocketException . . . . .	3521
6.749.2.4	SocketException . . . . .	3521
6.749.2.5	SocketException . . . . .	3522
6.749.2.6	SocketException . . . . .	3522
6.749.2.7	~SocketException . . . . .	3522
6.749.3	Member Function Documentation . . . . .	3522
6.749.3.1	clone . . . . .	3522
6.750	decaf::net::SocketFactory Class Reference . . . . .	3524
6.750.1	Detailed Description . . . . .	3525
6.750.2	Constructor & Destructor Documentation . . . . .	3525
6.750.2.1	SocketFactory . . . . .	3525
6.750.2.2	~SocketFactory . . . . .	3525
6.750.3	Member Function Documentation . . . . .	3525
6.750.3.1	createSocket . . . . .	3525

6.750.3.2 createSocket . . . . .	3525
6.750.3.3 createSocket . . . . .	3526
6.750.3.4 createSocket . . . . .	3526
6.750.3.5 createSocket . . . . .	3527
6.750.3.6 getDefault . . . . .	3527
6.751decaf::internal::net::SocketFileDescriptor Class Reference . . . . .	3528
6.751.1 Detailed Description . . . . .	3528
6.751.2 Constructor & Destructor Documentation . . . . .	3528
6.751.2.1 SocketFileDescriptor . . . . .	3528
6.751.2.2 ~SocketFileDescriptor . . . . .	3528
6.751.3 Member Function Documentation . . . . .	3528
6.751.3.1 getValue . . . . .	3528
6.752decaf::net::SocketImpl Class Reference . . . . .	3529
6.752.1 Detailed Description . . . . .	3531
6.752.2 Constructor & Destructor Documentation . . . . .	3531
6.752.2.1 SocketImpl . . . . .	3531
6.752.2.2 ~SocketImpl . . . . .	3531
6.752.3 Member Function Documentation . . . . .	3531
6.752.3.1 accept . . . . .	3531
6.752.3.2 available . . . . .	3531
6.752.3.3 bind . . . . .	3531
6.752.3.4 close . . . . .	3532
6.752.3.5 connect . . . . .	3532
6.752.3.6 create . . . . .	3532
6.752.3.7 getFileDescriptor . . . . .	3533
6.752.3.8 getInetAddress . . . . .	3533
6.752.3.9 getInputStream . . . . .	3533
6.752.3.10getLocalAddress . . . . .	3533
6.752.3.11getLocalPort . . . . .	3533
6.752.3.12getOption . . . . .	3534
6.752.3.13getOutputStream . . . . .	3534
6.752.3.14getPort . . . . .	3534
6.752.3.15listen . . . . .	3534
6.752.3.16sendUrgentData . . . . .	3535
6.752.3.17setOption . . . . .	3535
6.752.3.18shutdownInput . . . . .	3535



6.752.3.1	shutdownOutput . . . . .	3536
6.752.3.2	supportsUrgentData . . . . .	3536
6.752.3.2	toString . . . . .	3536
6.752.4	Field Documentation . . . . .	3536
6.752.4.1	address . . . . .	3536
6.752.4.2	fd . . . . .	3536
6.752.4.3	localPort . . . . .	3536
6.752.4.4	port . . . . .	3536
6.753	decaf::net::SocketImplFactory Class Reference . . . . .	3537
6.753.1	Detailed Description . . . . .	3537
6.753.2	Constructor & Destructor Documentation . . . . .	3537
6.753.2.1	~SocketImplFactory . . . . .	3537
6.753.3	Member Function Documentation . . . . .	3537
6.753.3.1	createSocketImpl . . . . .	3537
6.754	decaf::net::SocketOptions Class Reference . . . . .	3538
6.754.1	Detailed Description . . . . .	3539
6.754.2	Constructor & Destructor Documentation . . . . .	3539
6.754.2.1	~SocketOptions . . . . .	3539
6.754.3	Field Documentation . . . . .	3539
6.754.3.1	SOCKET_OPTION_BINDADDR . . . . .	3539
6.754.3.2	SOCKET_OPTION_BROADCAST . . . . .	3539
6.754.3.3	SOCKET_OPTION_IP_MULTICAST_IF . . . . .	3539
6.754.3.4	SOCKET_OPTION_IP_MULTICAST_IF2 . . . . .	3540
6.754.3.5	SOCKET_OPTION_IP_MULTICAST_LOOP . . . . .	3540
6.754.3.6	SOCKET_OPTION_IP_TOS . . . . .	3540
6.754.3.7	SOCKET_OPTION_KEEPALIVE . . . . .	3540
6.754.3.8	SOCKET_OPTION_LINGER . . . . .	3540
6.754.3.9	SOCKET_OPTION_OOBLINE . . . . .	3540
6.754.3.10	SOCKET_OPTION_RCVBUF . . . . .	3541
6.754.3.11	SOCKET_OPTION_REUSEADDR . . . . .	3541
6.754.3.12	SOCKET_OPTION_SNDBUF . . . . .	3541
6.754.3.13	SOCKET_OPTION_TCP_NODELAY . . . . .	3541
6.754.3.14	SOCKET_OPTION_TIMEOUT . . . . .	3541
6.755	decaf::net::SocketTimeoutException Class Reference . . . . .	3542
6.755.1	Constructor & Destructor Documentation . . . . .	3542
6.755.1.1	SocketTimeoutException . . . . .	3542

6.755.1.2 SocketTimeoutException . . . . .	3542
6.755.1.3 SocketTimeoutException . . . . .	3543
6.755.1.4 SocketTimeoutException . . . . .	3543
6.755.1.5 SocketTimeoutException . . . . .	3543
6.755.1.6 SocketTimeoutException . . . . .	3543
6.755.1.7 ~SocketTimeoutException . . . . .	3544
6.755.2 Member Function Documentation . . . . .	3544
6.755.2.1 clone . . . . .	3544
6.756decaf::net::ssl::SSLContext Class Reference . . . . .	3545
6.756.1 Detailed Description . . . . .	3545
6.756.2 Constructor & Destructor Documentation . . . . .	3545
6.756.2.1 SSLContext . . . . .	3545
6.756.2.2 ~SSLContext . . . . .	3545
6.756.3 Member Function Documentation . . . . .	3545
6.756.3.1 getDefault . . . . .	3545
6.756.3.2 getDefaultSSLParameters . . . . .	3546
6.756.3.3 getServerSocketFactory . . . . .	3546
6.756.3.4 getSocketFactory . . . . .	3546
6.756.3.5 getSupportedSSLParameters . . . . .	3546
6.756.3.6 setDefault . . . . .	3547
6.757decaf::net::ssl::SSLContextSpi Class Reference . . . . .	3548
6.757.1 Detailed Description . . . . .	3548
6.757.2 Constructor & Destructor Documentation . . . . .	3548
6.757.2.1 ~SSLContextSpi . . . . .	3548
6.757.3 Member Function Documentation . . . . .	3548
6.757.3.1 providerGetDefaultSSLParameters . . . . .	3548
6.757.3.2 providerGetServerSocketFactory . . . . .	3549
6.757.3.3 providerGetSocketFactory . . . . .	3549
6.757.3.4 providerGetSupportedSSLParameters . . . . .	3549
6.757.3.5 providerInit . . . . .	3550
6.758decaf::net::ssl::SSLParameters Class Reference . . . . .	3551
6.758.1 Constructor & Destructor Documentation . . . . .	3551
6.758.1.1 SSLParameters . . . . .	3551
6.758.1.2 SSLParameters . . . . .	3551
6.758.1.3 SSLParameters . . . . .	3552
6.758.1.4 ~SSLParameters . . . . .	3552

6.758.2 Member Function Documentation . . . . .	3552
6.758.2.1 getCipherSuites . . . . .	3552
6.758.2.2 getNeedClientAuth . . . . .	3552
6.758.2.3 getProtocols . . . . .	3552
6.758.2.4 getWantClientAuth . . . . .	3552
6.758.2.5 setCipherSuites . . . . .	3553
6.758.2.6 setNeedClientAuth . . . . .	3553
6.758.2.7 setProtocols . . . . .	3553
6.758.2.8 setWantClientAuth . . . . .	3553
6.759decaf::net::ssl::SSLServerSocket Class Reference . . . . .	3554
6.759.1 Detailed Description . . . . .	3555
6.759.2 Constructor & Destructor Documentation . . . . .	3555
6.759.2.1 SSLServerSocket . . . . .	3555
6.759.2.2 SSLServerSocket . . . . .	3555
6.759.2.3 SSLServerSocket . . . . .	3555
6.759.2.4 SSLServerSocket . . . . .	3556
6.759.2.5 ~SSLServerSocket . . . . .	3556
6.759.3 Member Function Documentation . . . . .	3556
6.759.3.1 getEnabledCipherSuites . . . . .	3556
6.759.3.2 getEnabledProtocols . . . . .	3557
6.759.3.3 getNeedClientAuth . . . . .	3557
6.759.3.4 getSupportedCipherSuites . . . . .	3557
6.759.3.5 getSupportedProtocols . . . . .	3557
6.759.3.6 getWantClientAuth . . . . .	3558
6.759.3.7 setEnabledCipherSuites . . . . .	3558
6.759.3.8 setEnabledProtocols . . . . .	3558
6.759.3.9 setNeedClientAuth . . . . .	3558
6.759.3.10set WantClientAuth . . . . .	3559
6.760decaf::net::ssl::SSLServerSocketFactory Class Reference . . . . .	3560
6.760.1 Detailed Description . . . . .	3560
6.760.2 Constructor & Destructor Documentation . . . . .	3561
6.760.2.1 SSLServerSocketFactory . . . . .	3561
6.760.2.2 ~SSLServerSocketFactory . . . . .	3561
6.760.3 Member Function Documentation . . . . .	3561
6.760.3.1 getDefault . . . . .	3561
6.760.3.2 getDefaultCipherSuites . . . . .	3561

6.760.3.3	getSupportedCipherSuites . . . . .	3561
6.761	decaf::net::ssl::SSLSocket Class Reference . . . . .	3563
6.761.1	Detailed Description . . . . .	3564
6.761.2	Constructor & Destructor Documentation . . . . .	3564
6.761.2.1	SSLSocket . . . . .	3564
6.761.2.2	SSLSocket . . . . .	3564
6.761.2.3	SSLSocket . . . . .	3565
6.761.2.4	SSLSocket . . . . .	3565
6.761.2.5	SSLSocket . . . . .	3566
6.761.2.6	~SSLSocket . . . . .	3566
6.761.3	Member Function Documentation . . . . .	3566
6.761.3.1	getEnabledCipherSuites . . . . .	3566
6.761.3.2	getEnabledProtocols . . . . .	3566
6.761.3.3	getNeedClientAuth . . . . .	3567
6.761.3.4	getSSLParameters . . . . .	3567
6.761.3.5	getSupportedCipherSuites . . . . .	3567
6.761.3.6	getSupportedProtocols . . . . .	3567
6.761.3.7	getUseClientMode . . . . .	3568
6.761.3.8	getWantClientAuth . . . . .	3568
6.761.3.9	setEnabledCipherSuites . . . . .	3568
6.761.3.10	setEnabledProtocols . . . . .	3568
6.761.3.11	setNeedClientAuth . . . . .	3569
6.761.3.12	setSSLParameters . . . . .	3569
6.761.3.13	setUseClientMode . . . . .	3569
6.761.3.14	setWantClientAuth . . . . .	3570
6.761.3.15	startHandshake . . . . .	3570
6.762	decaf::net::ssl::SSLSocketFactory Class Reference . . . . .	3571
6.762.1	Detailed Description . . . . .	3571
6.762.2	Constructor & Destructor Documentation . . . . .	3572
6.762.2.1	SSLSocketFactory . . . . .	3572
6.762.2.2	~SSLSocketFactory . . . . .	3572
6.762.3	Member Function Documentation . . . . .	3572
6.762.3.1	createSocket . . . . .	3572
6.762.3.2	getDefault . . . . .	3572
6.762.3.3	getDefaultCipherSuites . . . . .	3573
6.762.3.4	getSupportedCipherSuites . . . . .	3573

6.763activemq::transport::tcp::SslTransport Class Reference . . . . .	3574
6.763.1 Detailed Description . . . . .	3574
6.763.2 Constructor & Destructor Documentation . . . . .	3574
6.763.2.1 SslTransport . . . . .	3574
6.763.2.2 ~SslTransport . . . . .	3575
6.763.3 Member Function Documentation . . . . .	3575
6.763.3.1 configureSocket . . . . .	3575
6.763.3.2 createSocket . . . . .	3575
6.764activemq::transport::tcp::SslTransportFactory Class Reference . . . . .	3576
6.764.1 Constructor & Destructor Documentation . . . . .	3576
6.764.1.1 ~SslTransportFactory . . . . .	3576
6.764.2 Member Function Documentation . . . . .	3576
6.764.2.1 doCreateComposite . . . . .	3576
6.765activemq::commands::BrokerError::StackTraceElement Struct Reference . . . . .	3578
6.765.1 Field Documentation . . . . .	3578
6.765.1.1 ClassName . . . . .	3578
6.765.1.2 FileName . . . . .	3578
6.765.1.3 LineNumber . . . . .	3578
6.765.1.4 MethodName . . . . .	3578
6.766decaf::internal::io::StandardErrorOutputStream Class Reference . . . . .	3579
6.766.1 Detailed Description . . . . .	3579
6.766.2 Constructor & Destructor Documentation . . . . .	3580
6.766.2.1 StandardErrorOutputStream . . . . .	3580
6.766.2.2 ~StandardErrorOutputStream . . . . .	3580
6.766.3 Member Function Documentation . . . . .	3580
6.766.3.1 close . . . . .	3580
6.766.3.2 doWriteArrayBounded . . . . .	3580
6.766.3.3 doWriteByte . . . . .	3580
6.766.3.4 flush . . . . .	3580
6.767decaf::internal::io::StandardInputStream Class Reference . . . . .	3582
6.767.1 Constructor & Destructor Documentation . . . . .	3582
6.767.1.1 StandardInputStream . . . . .	3582
6.767.1.2 ~StandardInputStream . . . . .	3582
6.767.2 Member Function Documentation . . . . .	3582
6.767.2.1 available . . . . .	3582
6.767.2.2 doReadByte . . . . .	3583

6.768	decaf::internal::io::StandardOutputStream Class Reference . . . . .	3584
6.768.1	Constructor & Destructor Documentation . . . . .	3584
6.768.1.1	StandardOutputStream . . . . .	3584
6.768.1.2	~StandardOutputStream . . . . .	3584
6.768.2	Member Function Documentation . . . . .	3584
6.768.2.1	close . . . . .	3584
6.768.2.2	doWriteArrayBounded . . . . .	3585
6.768.2.3	doWriteByte . . . . .	3585
6.768.2.4	flush . . . . .	3585
6.769	cms::Startable Class Reference . . . . .	3586
6.769.1	Detailed Description . . . . .	3586
6.769.2	Constructor & Destructor Documentation . . . . .	3586
6.769.2.1	~Startable . . . . .	3586
6.769.3	Member Function Documentation . . . . .	3586
6.769.3.1	start . . . . .	3586
6.770	decaf::lang::STATIC_CAST_TOKEN Struct Reference . . . . .	3587
6.771	activemq::core::ActiveMQConstants::StaticInitializer Class Reference . . . . .	3588
6.771.1	Constructor & Destructor Documentation . . . . .	3588
6.771.1.1	StaticInitializer . . . . .	3588
6.771.1.2	~StaticInitializer . . . . .	3588
6.771.2	Field Documentation . . . . .	3588
6.771.2.1	destOptionMap . . . . .	3588
6.771.2.2	destOptions . . . . .	3588
6.771.2.3	uriParams . . . . .	3588
6.771.2.4	uriParamsMap . . . . .	3588
6.772	decaf::util::StlList< E > Class Template Reference . . . . .	3589
6.772.1	Detailed Description . . . . .	3593
6.772.2	Constructor & Destructor Documentation . . . . .	3593
6.772.2.1	StlList . . . . .	3593
6.772.2.2	StlList . . . . .	3594
6.772.2.3	StlList . . . . .	3594
6.772.2.4	~StlList . . . . .	3594
6.772.3	Member Function Documentation . . . . .	3594
6.772.3.1	add . . . . .	3594
6.772.3.2	add . . . . .	3594
6.772.3.3	addAll . . . . .	3595

6.772.3.4 clear . . . . .	3596
6.772.3.5 contains . . . . .	3596
6.772.3.6 copy . . . . .	3596
6.772.3.7 equals . . . . .	3597
6.772.3.8 get . . . . .	3597
6.772.3.9 indexOf . . . . .	3597
6.772.3.10isEmpty . . . . .	3598
6.772.3.11iterator . . . . .	3598
6.772.3.12iterator . . . . .	3598
6.772.3.13lastIndexOf . . . . .	3598
6.772.3.14listIterator . . . . .	3598
6.772.3.15listIterator . . . . .	3599
6.772.3.16listIterator . . . . .	3599
6.772.3.17listIterator . . . . .	3599
6.772.3.18remove . . . . .	3599
6.772.3.19remove . . . . .	3600
6.772.3.20set . . . . .	3600
6.772.3.21size . . . . .	3601
6.773decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference . . . . .	3602
6.773.1 Detailed Description . . . . .	3605
6.773.2 Constructor & Destructor Documentation . . . . .	3605
6.773.2.1 StlMap . . . . .	3605
6.773.2.2 StlMap . . . . .	3606
6.773.2.3 StlMap . . . . .	3606
6.773.2.4 ~StlMap . . . . .	3606
6.773.3 Member Function Documentation . . . . .	3606
6.773.3.1 clear . . . . .	3606
6.773.3.2 containsKey . . . . .	3606
6.773.3.3 containsValue . . . . .	3607
6.773.3.4 copy . . . . .	3607
6.773.3.5 copy . . . . .	3607
6.773.3.6 equals . . . . .	3608
6.773.3.7 equals . . . . .	3608
6.773.3.8 get . . . . .	3608
6.773.3.9 get . . . . .	3608
6.773.3.10isEmpty . . . . .	3609

6.773.3.1	keySet . . . . .	3609
6.773.3.12	lock . . . . .	3609
6.773.3.13	notify . . . . .	3610
6.773.3.14	notifyAll . . . . .	3610
6.773.3.15	put . . . . .	3610
6.773.3.16	putAll . . . . .	3611
6.773.3.17	putAll . . . . .	3611
6.773.3.18	remove . . . . .	3611
6.773.3.19	size . . . . .	3612
6.773.3.20	tryLock . . . . .	3612
6.773.3.21	unlock . . . . .	3612
6.773.3.22	values . . . . .	3612
6.773.3.23	wait . . . . .	3613
6.773.3.24	wait . . . . .	3613
6.773.3.25	wait . . . . .	3614
6.774	decaf::util::StlQueue< T > Class Template Reference . . . . .	3615
6.774.1	Detailed Description . . . . .	3616
6.774.2	Constructor & Destructor Documentation . . . . .	3617
6.774.2.1	StlQueue . . . . .	3617
6.774.2.2	~StlQueue . . . . .	3617
6.774.3	Member Function Documentation . . . . .	3617
6.774.3.1	back . . . . .	3617
6.774.3.2	back . . . . .	3617
6.774.3.3	clear . . . . .	3617
6.774.3.4	empty . . . . .	3617
6.774.3.5	enqueueFront . . . . .	3618
6.774.3.6	front . . . . .	3618
6.774.3.7	front . . . . .	3618
6.774.3.8	getSafeValue . . . . .	3618
6.774.3.9	iterator . . . . .	3618
6.774.3.10	lock . . . . .	3619
6.774.3.11	notify . . . . .	3619
6.774.3.12	notifyAll . . . . .	3619
6.774.3.13	pop . . . . .	3619
6.774.3.14	push . . . . .	3620
6.774.3.15	reverse . . . . .	3620



6.774.3.16	size . . . . .	3620
6.774.3.17	toArray . . . . .	3620
6.774.3.18	tryLock . . . . .	3620
6.774.3.19	unlock . . . . .	3621
6.774.3.20	wait . . . . .	3621
6.774.3.21	lwait . . . . .	3621
6.774.3.22	wait . . . . .	3622
6.775	decaf::util::StlSet< E > Class Template Reference . . . . .	3623
6.775.1	Detailed Description . . . . .	3625
6.775.2	Constructor & Destructor Documentation . . . . .	3625
6.775.2.1	StlSet . . . . .	3625
6.775.2.2	StlSet . . . . .	3625
6.775.2.3	StlSet . . . . .	3625
6.775.2.4	~StlSet . . . . .	3625
6.775.3	Member Function Documentation . . . . .	3625
6.775.3.1	add . . . . .	3625
6.775.3.2	clear . . . . .	3626
6.775.3.3	contains . . . . .	3627
6.775.3.4	copy . . . . .	3627
6.775.3.5	equals . . . . .	3627
6.775.3.6	isEmpty . . . . .	3627
6.775.3.7	iterator . . . . .	3627
6.775.3.8	iterator . . . . .	3628
6.775.3.9	remove . . . . .	3628
6.775.3.10	size . . . . .	3628
6.776	activemq::wireformat::stomp::StompCommandConstants Class Reference . . . . .	3629
6.776.1	Field Documentation . . . . .	3631
6.776.1.1	ABORT . . . . .	3631
6.776.1.2	ACK . . . . .	3631
6.776.1.3	ACK_AUTO . . . . .	3631
6.776.1.4	ACK_CLIENT . . . . .	3631
6.776.1.5	ACK_INDIVIDUAL . . . . .	3631
6.776.1.6	BEGIN . . . . .	3631
6.776.1.7	BYTES . . . . .	3631
6.776.1.8	COMMIT . . . . .	3631
6.776.1.9	CONNECT . . . . .	3631

6.776.1.10	CONNECTED	3631
6.776.1.11	DISCONNECT	3631
6.776.1.12	ERROR_CMD	3631
6.776.1.13	HEADER_ACK	3631
6.776.1.14	HEADER_CLIENT_ID	3631
6.776.1.15	HEADER_CONSUMERPRIORITY	3631
6.776.1.16	HEADER_CONTENTLENGTH	3631
6.776.1.17	HEADER_CORRELATIONID	3631
6.776.1.18	HEADER_DESTINATION	3631
6.776.1.19	HEADER_DISPATCH_ASYNC	3631
6.776.1.20	HEADER_EXCLUSIVE	3631
6.776.1.21	HEADER_EXPIRES	3631
6.776.1.22	HEADER_ID	3631
6.776.1.23	HEADER_JMSPRIORITY	3631
6.776.1.24	HEADER_LOGIN	3631
6.776.1.25	HEADER_MAXPENDINGMSGLIMIT	3631
6.776.1.26	HEADER_MESSAGE	3631
6.776.1.27	HEADER_MESSAGEID	3631
6.776.1.28	HEADER_NOLOCAL	3631
6.776.1.29	HEADER_OLDSUBSCRIPTIONNAME	3631
6.776.1.30	HEADER_PASSWORD	3631
6.776.1.31	HEADER_PERSISTENT	3631
6.776.1.32	HEADER_PREFETCHSIZE	3631
6.776.1.33	HEADER_RECEIPT_REQUIRED	3631
6.776.1.34	HEADER_RECEIPTID	3631
6.776.1.35	HEADER_REDELIVERED	3631
6.776.1.36	HEADER_REDELIVERYCOUNT	3631
6.776.1.37	HEADER_REPLYTO	3631
6.776.1.38	HEADER_REQUESTID	3631
6.776.1.39	HEADER_RESPONSEID	3631
6.776.1.40	HEADER_RETROACTIVE	3631
6.776.1.41	HEADER_SELECTOR	3631
6.776.1.42	HEADER_SESSIONID	3631
6.776.1.43	HEADER_SUBSCRIPTION	3631
6.776.1.44	HEADER_SUBSCRIPTIONNAME	3631
6.776.1.45	HEADER_TIMESTAMP	3631

6.776.1.46	HEADER_TRANSACTIONID . . . . .	3631
6.776.1.47	HEADER_TRANSFORMATION . . . . .	3631
6.776.1.48	HEADER_TRANSFORMATION_ERROR . . . . .	3631
6.776.1.49	HEADER_TYPE . . . . .	3631
6.776.1.50	MESSAGE . . . . .	3631
6.776.1.51	QUEUE_PREFIX . . . . .	3631
6.776.1.52	RECEIPT . . . . .	3631
6.776.1.53	SEND . . . . .	3631
6.776.1.54	SUBSCRIBE . . . . .	3631
6.776.1.55	TEMPQUEUE_PREFIX . . . . .	3631
6.776.1.56	TEMPTOPIC_PREFIX . . . . .	3631
6.776.1.57	TEXT . . . . .	3631
6.776.1.58	TOPIC_PREFIX . . . . .	3631
6.776.1.59	UNSUBSCRIBE . . . . .	3631
6.777	activemq::wireformat::stomp::StompFrame Class Reference . . . . .	3633
6.777.1	Detailed Description . . . . .	3634
6.777.2	Constructor & Destructor Documentation . . . . .	3634
6.777.2.1	StompFrame . . . . .	3634
6.777.2.2	~StompFrame . . . . .	3634
6.777.3	Member Function Documentation . . . . .	3634
6.777.3.1	clone . . . . .	3634
6.777.3.2	copy . . . . .	3634
6.777.3.3	fromStream . . . . .	3635
6.777.3.4	getBody . . . . .	3635
6.777.3.5	getBody . . . . .	3635
6.777.3.6	getBodyLength . . . . .	3635
6.777.3.7	getCommand . . . . .	3635
6.777.3.8	getProperties . . . . .	3635
6.777.3.9	getProperties . . . . .	3635
6.777.3.10	getProperty . . . . .	3636
6.777.3.11	hasProperty . . . . .	3636
6.777.3.12	removeProperty . . . . .	3636
6.777.3.13	setBody . . . . .	3636
6.777.3.14	setCommand . . . . .	3637
6.777.3.15	setProperty . . . . .	3637
6.777.3.16	oStream . . . . .	3637

6.778activemq::wireformat::stomp::StompHelper Class Reference . . . . .	3638
6.778.1 Detailed Description . . . . .	3639
6.778.2 Constructor & Destructor Documentation . . . . .	3639
6.778.2.1 StompHelper . . . . .	3639
6.778.2.2 ~StompHelper . . . . .	3639
6.778.3 Member Function Documentation . . . . .	3639
6.778.3.1 convertConsumerId . . . . .	3639
6.778.3.2 convertConsumerId . . . . .	3639
6.778.3.3 convertDestination . . . . .	3639
6.778.3.4 convertDestination . . . . .	3640
6.778.3.5 convertMessageId . . . . .	3640
6.778.3.6 convertMessageId . . . . .	3640
6.778.3.7 convertProducerId . . . . .	3640
6.778.3.8 convertProducerId . . . . .	3641
6.778.3.9 convertProperties . . . . .	3641
6.778.3.10convertProperties . . . . .	3641
6.778.3.11convertTransactionId . . . . .	3641
6.778.3.12convertTransactionId . . . . .	3642
6.779activemq::wireformat::stomp::StompWireFormat Class Reference . . . . .	3643
6.779.1 Constructor & Destructor Documentation . . . . .	3644
6.779.1.1 StompWireFormat . . . . .	3644
6.779.1.2 ~StompWireFormat . . . . .	3644
6.779.2 Member Function Documentation . . . . .	3644
6.779.2.1 createNegotiator . . . . .	3644
6.779.2.2 getVersion . . . . .	3644
6.779.2.3 hasNegotiator . . . . .	3644
6.779.2.4 inReceive . . . . .	3645
6.779.2.5 marshal . . . . .	3645
6.779.2.6 setVersion . . . . .	3645
6.779.2.7 unmarshal . . . . .	3645
6.780activemq::wireformat::stomp::StompWireFormatFactory Class Reference . . . . .	3647
6.780.1 Detailed Description . . . . .	3647
6.780.2 Constructor & Destructor Documentation . . . . .	3647
6.780.2.1 StompWireFormatFactory . . . . .	3647
6.780.2.2 ~StompWireFormatFactory . . . . .	3647
6.780.3 Member Function Documentation . . . . .	3647

6.780.3.1 createWireFormat . . . . .	3647
6.781cms::Stoppable Class Reference . . . . .	3648
6.781.1 Detailed Description . . . . .	3648
6.781.2 Constructor & Destructor Documentation . . . . .	3648
6.781.2.1 ~Stoppable . . . . .	3648
6.781.3 Member Function Documentation . . . . .	3648
6.781.3.1 stop . . . . .	3648
6.782decaf::util::logging::StreamHandler Class Reference . . . . .	3649
6.782.1 Detailed Description . . . . .	3649
6.782.2 Constructor & Destructor Documentation . . . . .	3650
6.782.2.1 StreamHandler . . . . .	3650
6.782.2.2 StreamHandler . . . . .	3650
6.782.2.3 ~StreamHandler . . . . .	3650
6.782.3 Member Function Documentation . . . . .	3650
6.782.3.1 close . . . . .	3650
6.782.3.2 close . . . . .	3650
6.782.3.3 flush . . . . .	3651
6.782.3.4 isLoggable . . . . .	3651
6.782.3.5 publish . . . . .	3651
6.782.3.6 setOutputStream . . . . .	3651
6.783cms::StreamMessage Class Reference . . . . .	3652
6.783.1 Detailed Description . . . . .	3654
6.783.2 Constructor & Destructor Documentation . . . . .	3655
6.783.2.1 ~StreamMessage . . . . .	3655
6.783.3 Member Function Documentation . . . . .	3655
6.783.3.1 readBoolean . . . . .	3655
6.783.3.2 readByte . . . . .	3655
6.783.3.3 readBytes . . . . .	3656
6.783.3.4 readBytes . . . . .	3656
6.783.3.5 readChar . . . . .	3657
6.783.3.6 readDouble . . . . .	3657
6.783.3.7 readFloat . . . . .	3658
6.783.3.8 readInt . . . . .	3658
6.783.3.9 readLong . . . . .	3658
6.783.3.10readShort . . . . .	3659
6.783.3.11readString . . . . .	3659

6.783.3.12	readUnsignedShort . . . . .	3660
6.783.3.13	writeBoolean . . . . .	3660
6.783.3.14	writeByte . . . . .	3660
6.783.3.15	writeBytes . . . . .	3661
6.783.3.16	writeBytes . . . . .	3661
6.783.3.17	writeChar . . . . .	3661
6.783.3.18	writeDouble . . . . .	3662
6.783.3.19	writeFloat . . . . .	3662
6.783.3.20	writeInt . . . . .	3662
6.783.3.21	writeLong . . . . .	3663
6.783.3.22	writeShort . . . . .	3663
6.783.3.23	writeString . . . . .	3663
6.783.3.24	writeUnsignedShort . . . . .	3664
6.784	decaf::lang::String Class Reference . . . . .	3665
6.784.1	Detailed Description . . . . .	3666
6.784.2	Constructor & Destructor Documentation . . . . .	3666
6.784.2.1	String . . . . .	3666
6.784.2.2	String . . . . .	3666
6.784.2.3	~String . . . . .	3666
6.784.3	Member Function Documentation . . . . .	3666
6.784.3.1	charAt . . . . .	3666
6.784.3.2	isEmpty . . . . .	3667
6.784.3.3	length . . . . .	3667
6.784.3.4	subSequence . . . . .	3667
6.784.3.5	toString . . . . .	3667
6.785	decaf::util::StringTokenizer Class Reference . . . . .	3668
6.785.1	Constructor & Destructor Documentation . . . . .	3668
6.785.1.1	StringTokenizer . . . . .	3668
6.785.1.2	~StringTokenizer . . . . .	3669
6.785.2	Member Function Documentation . . . . .	3669
6.785.2.1	countTokens . . . . .	3669
6.785.2.2	hasMoreTokens . . . . .	3669
6.785.2.3	nextToken . . . . .	3669
6.785.2.4	nextToken . . . . .	3669
6.785.2.5	reset . . . . .	3670
6.785.2.6	toArray . . . . .	3670

6.786	activemq::commands::SubscriptionInfo Class Reference . . . . .	3671
6.786.1	Constructor & Destructor Documentation . . . . .	3672
6.786.1.1	SubscriptionInfo . . . . .	3672
6.786.1.2	~SubscriptionInfo . . . . .	3672
6.786.2	Member Function Documentation . . . . .	3672
6.786.2.1	cloneDataStructure . . . . .	3672
6.786.2.2	copyDataStructure . . . . .	3672
6.786.2.3	equals . . . . .	3672
6.786.2.4	getClientId . . . . .	3673
6.786.2.5	getClientId . . . . .	3673
6.786.2.6	getDataStructureType . . . . .	3673
6.786.2.7	getDestination . . . . .	3674
6.786.2.8	getDestination . . . . .	3674
6.786.2.9	getSelector . . . . .	3674
6.786.2.10	getSelector . . . . .	3674
6.786.2.11	getSubscriptionName . . . . .	3674
6.786.2.12	getSubscriptionName . . . . .	3674
6.786.2.13	getSubscribedDestination . . . . .	3674
6.786.2.14	getSubscribedDestination . . . . .	3674
6.786.2.15	setClientId . . . . .	3674
6.786.2.16	setDestination . . . . .	3674
6.786.2.17	setSelector . . . . .	3674
6.786.2.18	setSubscriptionName . . . . .	3674
6.786.2.19	setSubscribedDestination . . . . .	3674
6.786.2.20	toString . . . . .	3674
6.786.3	Field Documentation . . . . .	3675
6.786.3.1	clientId . . . . .	3675
6.786.3.2	destination . . . . .	3675
6.786.3.3	ID_SUBSCRIPTIONINFO . . . . .	3675
6.786.3.4	selector . . . . .	3675
6.786.3.5	subscriptionName . . . . .	3675
6.786.3.6	subscribedDestination . . . . .	3675
6.787	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference . . . . .	3676
6.787.1	Detailed Description . . . . .	3676
6.787.2	Constructor & Destructor Documentation . . . . .	3677
6.787.2.1	SubscriptionInfoMarshaller . . . . .	3677

6.787.2.2 ~SubscriptionInfoMarshaller . . . . .	3677
6.787.3 Member Function Documentation . . . . .	3677
6.787.3.1 createObject . . . . .	3677
6.787.3.2 getDataStructureType . . . . .	3677
6.787.3.3 looseMarshal . . . . .	3677
6.787.3.4 looseUnmarshal . . . . .	3678
6.787.3.5 tightMarshal1 . . . . .	3678
6.787.3.6 tightMarshal2 . . . . .	3678
6.787.3.7 tightUnmarshal . . . . .	3679
6.788activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class	
Reference . . . . .	3680
6.788.1 Detailed Description . . . . .	3680
6.788.2 Constructor & Destructor Documentation . . . . .	3681
6.788.2.1 SubscriptionInfoMarshaller . . . . .	3681
6.788.2.2 ~SubscriptionInfoMarshaller . . . . .	3681
6.788.3 Member Function Documentation . . . . .	3681
6.788.3.1 createObject . . . . .	3681
6.788.3.2 getDataStructureType . . . . .	3681
6.788.3.3 looseMarshal . . . . .	3681
6.788.3.4 looseUnmarshal . . . . .	3682
6.788.3.5 tightMarshal1 . . . . .	3682
6.788.3.6 tightMarshal2 . . . . .	3682
6.788.3.7 tightUnmarshal . . . . .	3683
6.789activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class	
Reference . . . . .	3684
6.789.1 Detailed Description . . . . .	3684
6.789.2 Constructor & Destructor Documentation . . . . .	3685
6.789.2.1 SubscriptionInfoMarshaller . . . . .	3685
6.789.2.2 ~SubscriptionInfoMarshaller . . . . .	3685
6.789.3 Member Function Documentation . . . . .	3685
6.789.3.1 createObject . . . . .	3685
6.789.3.2 getDataStructureType . . . . .	3685
6.789.3.3 looseMarshal . . . . .	3685
6.789.3.4 looseUnmarshal . . . . .	3686
6.789.3.5 tightMarshal1 . . . . .	3686
6.789.3.6 tightMarshal2 . . . . .	3686
6.789.3.7 tightUnmarshal . . . . .	3687



6.790	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	Class	
	Reference		3688
6.790.1	Detailed Description		3688
6.790.2	Constructor & Destructor Documentation		3689
	6.790.2.1 SubscriptionInfoMarshaller		3689
	6.790.2.2 ~SubscriptionInfoMarshaller		3689
6.790.3	Member Function Documentation		3689
	6.790.3.1 createObject		3689
	6.790.3.2 getDataStructureType		3689
	6.790.3.3 looseMarshal		3689
	6.790.3.4 looseUnmarshal		3690
	6.790.3.5 tightMarshal1		3690
	6.790.3.6 tightMarshal2		3690
	6.790.3.7 tightUnmarshal		3691
6.791	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	Class	
	Reference		3692
6.791.1	Detailed Description		3692
6.791.2	Constructor & Destructor Documentation		3693
	6.791.2.1 SubscriptionInfoMarshaller		3693
	6.791.2.2 ~SubscriptionInfoMarshaller		3693
6.791.3	Member Function Documentation		3693
	6.791.3.1 createObject		3693
	6.791.3.2 getDataStructureType		3693
	6.791.3.3 looseMarshal		3693
	6.791.3.4 looseUnmarshal		3694
	6.791.3.5 tightMarshal1		3694
	6.791.3.6 tightMarshal2		3694
	6.791.3.7 tightUnmarshal		3695
6.792	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	Class	
	Reference		3696
6.792.1	Detailed Description		3696
6.792.2	Constructor & Destructor Documentation		3697
	6.792.2.1 SubscriptionInfoMarshaller		3697
	6.792.2.2 ~SubscriptionInfoMarshaller		3697
6.792.3	Member Function Documentation		3697
	6.792.3.1 createObject		3697
	6.792.3.2 getDataStructureType		3697

6.792.3.3 looseMarshal . . . . .	3697
6.792.3.4 looseUnmarshal . . . . .	3698
6.792.3.5 tightMarshal1 . . . . .	3698
6.792.3.6 tightMarshal2 . . . . .	3698
6.792.3.7 tightUnmarshal . . . . .	3699
6.793decaf::util::concurrent::Synchronizable Class Reference . . . . .	3700
6.793.1 Detailed Description . . . . .	3701
6.793.2 Constructor & Destructor Documentation . . . . .	3701
6.793.2.1 ~Synchronizable . . . . .	3701
6.793.3 Member Function Documentation . . . . .	3701
6.793.3.1 lock . . . . .	3701
6.793.3.2 notify . . . . .	3702
6.793.3.3 notifyAll . . . . .	3703
6.793.3.4 tryLock . . . . .	3704
6.793.3.5 unlock . . . . .	3705
6.793.3.6 wait . . . . .	3707
6.793.3.7 wait . . . . .	3708
6.793.3.8 wait . . . . .	3709
6.794decaf::internal::util::concurrent::SynchronizableImpl Class Reference . . . . .	3711
6.794.1 Detailed Description . . . . .	3712
6.794.2 Constructor & Destructor Documentation . . . . .	3712
6.794.2.1 SynchronizableImpl . . . . .	3712
6.794.2.2 ~SynchronizableImpl . . . . .	3712
6.794.3 Member Function Documentation . . . . .	3712
6.794.3.1 lock . . . . .	3712
6.794.3.2 notify . . . . .	3712
6.794.3.3 notifyAll . . . . .	3712
6.794.3.4 tryLock . . . . .	3713
6.794.3.5 unlock . . . . .	3713
6.794.3.6 wait . . . . .	3713
6.794.3.7 wait . . . . .	3714
6.794.3.8 wait . . . . .	3714
6.795activemq::core::Synchronization Class Reference . . . . .	3715
6.795.1 Detailed Description . . . . .	3715
6.795.2 Constructor & Destructor Documentation . . . . .	3715
6.795.2.1 ~Synchronization . . . . .	3715

6.795.3 Member Function Documentation . . . . .	3715
6.795.3.1 afterCommit . . . . .	3715
6.795.3.2 afterRollback . . . . .	3715
6.795.3.3 beforeEnd . . . . .	3715
6.796decaf::util::concurrent::SynchronousQueue< E > Class Template Reference . . . .	3716
6.796.1 Detailed Description . . . . .	3717
6.796.2 Constructor & Destructor Documentation . . . . .	3718
6.796.2.1 SynchronousQueue . . . . .	3718
6.796.2.2 ~SynchronousQueue . . . . .	3718
6.796.3 Member Function Documentation . . . . .	3718
6.796.3.1 clear . . . . .	3718
6.796.3.2 contains . . . . .	3719
6.796.3.3 containsAll . . . . .	3719
6.796.3.4 drainTo . . . . .	3719
6.796.3.5 drainTo . . . . .	3720
6.796.3.6 equals . . . . .	3720
6.796.3.7 isEmpty . . . . .	3721
6.796.3.8 iterator . . . . .	3721
6.796.3.9 iterator . . . . .	3721
6.796.3.10offer . . . . .	3721
6.796.3.11offer . . . . .	3722
6.796.3.12peek . . . . .	3722
6.796.3.13poll . . . . .	3722
6.796.3.14poll . . . . .	3722
6.796.3.15put . . . . .	3723
6.796.3.16remainingCapacity . . . . .	3723
6.796.3.17remove . . . . .	3724
6.796.3.18removeAll . . . . .	3724
6.796.3.19retainAll . . . . .	3724
6.796.3.20size . . . . .	3724
6.796.3.21take . . . . .	3724
6.796.3.22toArray . . . . .	3724
6.797decaf::lang::System Class Reference . . . . .	3726
6.797.1 Detailed Description . . . . .	3727
6.797.2 Constructor & Destructor Documentation . . . . .	3728
6.797.2.1 System . . . . .	3728

6.797.2.2	~System . . . . .	3728
6.797.3	Member Function Documentation . . . . .	3728
6.797.3.1	arraycopy . . . . .	3728
6.797.3.2	arraycopy . . . . .	3728
6.797.3.3	arraycopy . . . . .	3729
6.797.3.4	arraycopy . . . . .	3729
6.797.3.5	availableProcessors . . . . .	3729
6.797.3.6	clearProperty . . . . .	3730
6.797.3.7	currentTimeMillis . . . . .	3730
6.797.3.8	getenv . . . . .	3730
6.797.3.9	getenv . . . . .	3730
6.797.3.10	getProperties . . . . .	3731
6.797.3.11	getProperty . . . . .	3731
6.797.3.12	getProperty . . . . .	3731
6.797.3.13	nanoTime . . . . .	3732
6.797.3.14	setenv . . . . .	3732
6.797.3.15	setProperty . . . . .	3732
6.797.3.16	unsetenv . . . . .	3733
6.797.4	Friends And Related Function Documentation . . . . .	3733
6.797.4.1	decaf::lang::Runtime . . . . .	3733
6.798	activemq::threads::Task Class Reference . . . . .	3734
6.798.1	Detailed Description . . . . .	3734
6.798.2	Constructor & Destructor Documentation . . . . .	3734
6.798.2.1	~Task . . . . .	3734
6.798.3	Member Function Documentation . . . . .	3734
6.798.3.1	iterate . . . . .	3734
6.799	decaf::util::concurrent::TaskListener Class Reference . . . . .	3735
6.799.1	Constructor & Destructor Documentation . . . . .	3735
6.799.1.1	~TaskListener . . . . .	3735
6.799.2	Member Function Documentation . . . . .	3735
6.799.2.1	onTaskComplete . . . . .	3735
6.799.2.2	onTaskException . . . . .	3735
6.800	activemq::threads::TaskRunner Class Reference . . . . .	3736
6.800.1	Constructor & Destructor Documentation . . . . .	3736
6.800.1.1	~TaskRunner . . . . .	3736
6.800.2	Member Function Documentation . . . . .	3736

6.800.2.1 shutdown . . . . .	3736
6.800.2.2 shutdown . . . . .	3736
6.800.2.3 wakeup . . . . .	3737
6.801decaf::internal::net::tcp::TcpSocket Class Reference . . . . .	3738
6.801.1 Detailed Description . . . . .	3740
6.801.2 Constructor & Destructor Documentation . . . . .	3741
6.801.2.1 TcpSocket . . . . .	3741
6.801.2.2 ~TcpSocket . . . . .	3741
6.801.3 Member Function Documentation . . . . .	3741
6.801.3.1 accept . . . . .	3741
6.801.3.2 available . . . . .	3741
6.801.3.3 bind . . . . .	3741
6.801.3.4 checkResult . . . . .	3742
6.801.3.5 close . . . . .	3742
6.801.3.6 connect . . . . .	3742
6.801.3.7 create . . . . .	3742
6.801.3.8 getInputStream . . . . .	3743
6.801.3.9 getLocalAddress . . . . .	3743
6.801.3.10getOption . . . . .	3743
6.801.3.11getOutputStream . . . . .	3743
6.801.3.12getSocketHandle . . . . .	3744
6.801.3.13sClosed . . . . .	3744
6.801.3.14sConnected . . . . .	3744
6.801.3.15listen . . . . .	3744
6.801.3.16read . . . . .	3745
6.801.3.17setOption . . . . .	3745
6.801.3.18shutdownInput . . . . .	3745
6.801.3.19shutdownOutput . . . . .	3746
6.801.3.20write . . . . .	3746
6.802decaf::internal::net::tcp::TcpSocketInputStream Class Reference . . . . .	3747
6.802.1 Detailed Description . . . . .	3747
6.802.2 Constructor & Destructor Documentation . . . . .	3748
6.802.2.1 TcpSocketInputStream . . . . .	3748
6.802.2.2 ~TcpSocketInputStream . . . . .	3748
6.802.3 Member Function Documentation . . . . .	3748
6.802.3.1 available . . . . .	3748

6.802.3.2 close . . . . .	3748
6.802.3.3 doReadArrayBounded . . . . .	3749
6.802.3.4 doReadByte . . . . .	3749
6.802.3.5 skip . . . . .	3749
6.803decaf::internal::net::tcp::TcpSocketOutputStream Class Reference . . . . .	3750
6.803.1 Detailed Description . . . . .	3750
6.803.2 Constructor & Destructor Documentation . . . . .	3750
6.803.2.1 TcpSocketOutputStream . . . . .	3750
6.803.2.2 ~TcpSocketOutputStream . . . . .	3751
6.803.3 Member Function Documentation . . . . .	3751
6.803.3.1 close . . . . .	3751
6.803.3.2 doWriteArrayBounded . . . . .	3751
6.803.3.3 doWriteByte . . . . .	3751
6.804activemq::transport::tcp::TcpTransport Class Reference . . . . .	3752
6.804.1 Detailed Description . . . . .	3753
6.804.2 Constructor & Destructor Documentation . . . . .	3753
6.804.2.1 TcpTransport . . . . .	3753
6.804.2.2 ~TcpTransport . . . . .	3753
6.804.3 Member Function Documentation . . . . .	3753
6.804.3.1 close . . . . .	3753
6.804.3.2 configureSocket . . . . .	3753
6.804.3.3 connect . . . . .	3754
6.804.3.4 createSocket . . . . .	3754
6.804.3.5 isClosed . . . . .	3754
6.804.3.6 isConnected . . . . .	3754
6.804.3.7 isFaultTolerant . . . . .	3755
6.805activemq::transport::tcp::TcpTransportFactory Class Reference . . . . .	3756
6.805.1 Detailed Description . . . . .	3756
6.805.2 Constructor & Destructor Documentation . . . . .	3757
6.805.2.1 ~TcpTransportFactory . . . . .	3757
6.805.3 Member Function Documentation . . . . .	3757
6.805.3.1 create . . . . .	3757
6.805.3.2 createComposite . . . . .	3757
6.805.3.3 doCreateComposite . . . . .	3757
6.806cms::TemporaryQueue Class Reference . . . . .	3759
6.806.1 Detailed Description . . . . .	3759

6.806.2 Constructor & Destructor Documentation . . . . .	3759
6.806.2.1 ~TemporaryQueue . . . . .	3759
6.806.3 Member Function Documentation . . . . .	3759
6.806.3.1 destroy . . . . .	3759
6.806.3.2 getQueueName . . . . .	3760
6.807cms::TemporaryTopic Class Reference . . . . .	3761
6.807.1 Detailed Description . . . . .	3761
6.807.2 Constructor & Destructor Documentation . . . . .	3761
6.807.2.1 ~TemporaryTopic . . . . .	3761
6.807.3 Member Function Documentation . . . . .	3761
6.807.3.1 destroy . . . . .	3761
6.807.3.2 getTopicName . . . . .	3762
6.808cms::TextMessage Class Reference . . . . .	3763
6.808.1 Detailed Description . . . . .	3763
6.808.2 Constructor & Destructor Documentation . . . . .	3763
6.808.2.1 ~TextMessage . . . . .	3763
6.808.3 Member Function Documentation . . . . .	3763
6.808.3.1 getText . . . . .	3763
6.808.3.2 setText . . . . .	3764
6.808.3.3 setText . . . . .	3764
6.809decaf::lang::Thread Class Reference . . . . .	3765
6.809.1 Detailed Description . . . . .	3767
6.809.2 Member Enumeration Documentation . . . . .	3768
6.809.2.1 State . . . . .	3768
6.809.3 Constructor & Destructor Documentation . . . . .	3768
6.809.3.1 Thread . . . . .	3768
6.809.3.2 Thread . . . . .	3768
6.809.3.3 Thread . . . . .	3768
6.809.3.4 Thread . . . . .	3769
6.809.3.5 ~Thread . . . . .	3769
6.809.4 Member Function Documentation . . . . .	3769
6.809.4.1 currentThread . . . . .	3769
6.809.4.2 getId . . . . .	3769
6.809.4.3 getName . . . . .	3769
6.809.4.4 getPriority . . . . .	3769
6.809.4.5 getState . . . . .	3770

6.809.4.6	getUncaughtExceptionHandler . . . . .	3770
6.809.4.7	isAlive . . . . .	3770
6.809.4.8	join . . . . .	3770
6.809.4.9	join . . . . .	3770
6.809.4.10	join . . . . .	3771
6.809.4.11	run . . . . .	3771
6.809.4.12	setName . . . . .	3771
6.809.4.13	setPriority . . . . .	3771
6.809.4.14	setUncaughtExceptionHandler . . . . .	3772
6.809.4.15	sleep . . . . .	3772
6.809.4.16	sleep . . . . .	3772
6.809.4.17	start . . . . .	3772
6.809.4.18	toString . . . . .	3773
6.809.4.19	yield . . . . .	3773
6.809.5	Friends And Related Function Documentation . . . . .	3773
6.809.5.1	decaf::lang::Runtime . . . . .	3773
6.809.5.2	decaf::util::concurrent::locks::LockSupport . . . . .	3773
6.809.6	Field Documentation . . . . .	3773
6.809.6.1	MAX_PRIORITY . . . . .	3773
6.809.6.2	MIN_PRIORITY . . . . .	3773
6.809.6.3	NORM_PRIORITY . . . . .	3773
6.810	decaf::util::concurrent::ThreadFactory Class Reference . . . . .	3774
6.810.1	Detailed Description . . . . .	3774
6.810.2	Constructor & Destructor Documentation . . . . .	3774
6.810.2.1	~ThreadFactory . . . . .	3774
6.810.3	Member Function Documentation . . . . .	3774
6.810.3.1	newThread . . . . .	3774
6.811	decaf::lang::ThreadGroup Class Reference . . . . .	3776
6.811.1	Detailed Description . . . . .	3776
6.811.2	Constructor & Destructor Documentation . . . . .	3776
6.811.2.1	ThreadGroup . . . . .	3776
6.811.2.2	~ThreadGroup . . . . .	3776
6.812	decaf::util::concurrent::ThreadPool Class Reference . . . . .	3777
6.812.1	Detailed Description . . . . .	3778
6.812.2	Member Typedef Documentation . . . . .	3779
6.812.2.1	Task . . . . .	3779



6.812.3 Constructor & Destructor Documentation . . . . .	3779
6.812.3.1 ThreadPool . . . . .	3779
6.812.3.2 ~ThreadPool . . . . .	3779
6.812.4 Member Function Documentation . . . . .	3779
6.812.4.1 deQueueTask . . . . .	3779
6.812.4.2 getBacklog . . . . .	3779
6.812.4.3 getBlockSize . . . . .	3779
6.812.4.4 getFreeThreadCount . . . . .	3779
6.812.4.5 getInstance . . . . .	3780
6.812.4.6 getMaxThreads . . . . .	3780
6.812.4.7 getPoolSize . . . . .	3780
6.812.4.8 onTaskCompleted . . . . .	3780
6.812.4.9 onTaskException . . . . .	3780
6.812.4.10 onTaskStarted . . . . .	3781
6.812.4.11 queueTask . . . . .	3781
6.812.4.12 reserve . . . . .	3781
6.812.4.13 setBlockSize . . . . .	3781
6.812.4.14 setMaxThreads . . . . .	3782
6.812.5 Field Documentation . . . . .	3782
6.812.5.1 DEFAULT_MAX_BLOCK_SIZE . . . . .	3782
6.812.5.2 DEFAULT_MAX_POOL_SIZE . . . . .	3782
6.813 decaf::lang::Throwable Class Reference . . . . .	3783
6.813.1 Detailed Description . . . . .	3783
6.813.2 Constructor & Destructor Documentation . . . . .	3784
6.813.2.1 Throwable . . . . .	3784
6.813.2.2 ~Throwable . . . . .	3784
6.813.3 Member Function Documentation . . . . .	3784
6.813.3.1 clone . . . . .	3784
6.813.3.2 getCause . . . . .	3785
6.813.3.3 getMessage . . . . .	3785
6.813.3.4 getStackTrace . . . . .	3785
6.813.3.5 getStackTraceString . . . . .	3786
6.813.3.6 initCause . . . . .	3786
6.813.3.7 printStackTrace . . . . .	3786
6.813.3.8 printStackTrace . . . . .	3786
6.813.3.9 setMark . . . . .	3786

6.814decaf::util::concurrent::TimeoutException Class Reference . . . . .	3787
6.814.1 Constructor & Destructor Documentation . . . . .	3787
6.814.1.1 TimeoutException . . . . .	3787
6.814.1.2 TimeoutException . . . . .	3787
6.814.1.3 TimeoutException . . . . .	3788
6.814.1.4 TimeoutException . . . . .	3788
6.814.1.5 TimeoutException . . . . .	3788
6.814.1.6 TimeoutException . . . . .	3788
6.814.1.7 ~TimeoutException . . . . .	3789
6.814.2 Member Function Documentation . . . . .	3789
6.814.2.1 clone . . . . .	3789
6.815decaf::util::Timer Class Reference . . . . .	3790
6.815.1 Detailed Description . . . . .	3791
6.815.2 Constructor & Destructor Documentation . . . . .	3792
6.815.2.1 Timer . . . . .	3792
6.815.2.2 ~Timer . . . . .	3792
6.815.3 Member Function Documentation . . . . .	3792
6.815.3.1 cancel . . . . .	3792
6.815.3.2 purge . . . . .	3792
6.815.3.3 schedule . . . . .	3792
6.815.3.4 schedule . . . . .	3793
6.815.3.5 schedule . . . . .	3794
6.815.3.6 schedule . . . . .	3794
6.815.3.7 schedule . . . . .	3795
6.815.3.8 schedule . . . . .	3796
6.815.3.9 schedule . . . . .	3796
6.815.3.10 schedule . . . . .	3797
6.815.3.11 scheduleAtFixedRate . . . . .	3797
6.815.3.12 scheduleAtFixedRate . . . . .	3798
6.815.3.13 scheduleAtFixedRate . . . . .	3799
6.815.3.14 scheduleAtFixedRate . . . . .	3799
6.816decaf::util::TimerTask Class Reference . . . . .	3801
6.816.1 Detailed Description . . . . .	3801
6.816.2 Constructor & Destructor Documentation . . . . .	3802
6.816.2.1 TimerTask . . . . .	3802
6.816.2.2 ~TimerTask . . . . .	3802

6.816.3 Member Function Documentation . . . . .	3802
6.816.3.1 cancel . . . . .	3802
6.816.3.2 getWhen . . . . .	3802
6.816.3.3 isScheduled . . . . .	3802
6.816.3.4 scheduledExecutionTime . . . . .	3802
6.816.3.5 setScheduledTime . . . . .	3803
6.816.4 Friends And Related Function Documentation . . . . .	3803
6.816.4.1 decaf::internal::util::TimerTaskHeap . . . . .	3803
6.816.4.2 Timer . . . . .	3803
6.816.4.3 TimerImpl . . . . .	3803
6.817 decaf::internal::util::TimerTaskHeap Class Reference . . . . .	3804
6.817.1 Detailed Description . . . . .	3804
6.817.2 Constructor & Destructor Documentation . . . . .	3805
6.817.2.1 TimerTaskHeap . . . . .	3805
6.817.2.2 ~TimerTaskHeap . . . . .	3805
6.817.3 Member Function Documentation . . . . .	3805
6.817.3.1 adjustMinimum . . . . .	3805
6.817.3.2 deleteIfCancelled . . . . .	3805
6.817.3.3 find . . . . .	3805
6.817.3.4 insert . . . . .	3805
6.817.3.5 isEmpty . . . . .	3805
6.817.3.6 peek . . . . .	3806
6.817.3.7 remove . . . . .	3806
6.817.3.8 reset . . . . .	3806
6.817.3.9 size . . . . .	3806
6.818 decaf::util::concurrent::TimeUnit Class Reference . . . . .	3807
6.818.1 Detailed Description . . . . .	3808
6.818.2 Constructor & Destructor Documentation . . . . .	3809
6.818.2.1 TimeUnit . . . . .	3809
6.818.2.2 ~TimeUnit . . . . .	3809
6.818.3 Member Function Documentation . . . . .	3809
6.818.3.1 compareTo . . . . .	3809
6.818.3.2 convert . . . . .	3810
6.818.3.3 equals . . . . .	3810
6.818.3.4 operator< . . . . .	3810
6.818.3.5 operator== . . . . .	3811

6.818.3.6 sleep . . . . .	3811
6.818.3.7 timedJoin . . . . .	3811
6.818.3.8 timedWait . . . . .	3812
6.818.3.9 toDays . . . . .	3812
6.818.3.10 toHours . . . . .	3813
6.818.3.11 toMicros . . . . .	3813
6.818.3.12 toMillis . . . . .	3813
6.818.3.13 toMinutes . . . . .	3814
6.818.3.14 toNanos . . . . .	3814
6.818.3.15 toSeconds . . . . .	3814
6.818.3.16 toString . . . . .	3815
6.818.3.17 valueOf . . . . .	3815
6.818.4 Field Documentation . . . . .	3815
6.818.4.1 DAYS . . . . .	3815
6.818.4.2 HOURS . . . . .	3815
6.818.4.3 MICROSECONDS . . . . .	3815
6.818.4.4 MILLISECONDS . . . . .	3815
6.818.4.5 MINUTES . . . . .	3815
6.818.4.6 NANOSECONDS . . . . .	3815
6.818.4.7 SECONDS . . . . .	3816
6.818.4.8 values . . . . .	3816
6.819 cms::Topic Class Reference . . . . .	3817
6.819.1 Detailed Description . . . . .	3817
6.819.2 Constructor & Destructor Documentation . . . . .	3817
6.819.2.1 ~Topic . . . . .	3817
6.819.3 Member Function Documentation . . . . .	3817
6.819.3.1 getTopicName . . . . .	3817
6.820 activemq::state::Tracked Class Reference . . . . .	3818
6.820.1 Constructor & Destructor Documentation . . . . .	3818
6.820.1.1 Tracked . . . . .	3818
6.820.1.2 Tracked . . . . .	3818
6.820.1.3 ~Tracked . . . . .	3818
6.820.2 Member Function Documentation . . . . .	3818
6.820.2.1 isWaitingForResponse . . . . .	3818
6.820.2.2 onResponse . . . . .	3818
6.821 activemq::commands::TransactionId Class Reference . . . . .	3819

6.821.1 Member Typedef Documentation . . . . .	3819
6.821.1.1 COMPARATOR . . . . .	3819
6.821.2 Constructor & Destructor Documentation . . . . .	3820
6.821.2.1 TransactionId . . . . .	3820
6.821.2.2 TransactionId . . . . .	3820
6.821.2.3 ~TransactionId . . . . .	3820
6.821.3 Member Function Documentation . . . . .	3820
6.821.3.1 cloneDataStructure . . . . .	3820
6.821.3.2 compareTo . . . . .	3820
6.821.3.3 copyDataStructure . . . . .	3820
6.821.3.4 equals . . . . .	3820
6.821.3.5 equals . . . . .	3821
6.821.3.6 getDataStructureType . . . . .	3821
6.821.3.7 operator< . . . . .	3821
6.821.3.8 operator= . . . . .	3821
6.821.3.9 operator== . . . . .	3821
6.821.3.10 toString . . . . .	3821
6.821.4 Field Documentation . . . . .	3822
6.821.4.1 ID_TRANSACTIONID . . . . .	3822
6.822activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference . . . . .	3823
6.822.1 Detailed Description . . . . .	3823
6.822.2 Constructor & Destructor Documentation . . . . .	3824
6.822.2.1 TransactionIdMarshaller . . . . .	3824
6.822.2.2 ~TransactionIdMarshaller . . . . .	3824
6.822.3 Member Function Documentation . . . . .	3824
6.822.3.1 looseMarshal . . . . .	3824
6.822.3.2 looseUnmarshal . . . . .	3824
6.822.3.3 tightMarshal1 . . . . .	3825
6.822.3.4 tightMarshal2 . . . . .	3825
6.822.3.5 tightUnmarshal . . . . .	3826
6.823activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference . . . . .	3827
6.823.1 Detailed Description . . . . .	3827
6.823.2 Constructor & Destructor Documentation . . . . .	3828
6.823.2.1 TransactionIdMarshaller . . . . .	3828
6.823.2.2 ~TransactionIdMarshaller . . . . .	3828

6.823.3 Member Function Documentation . . . . .	3828
6.823.3.1 looseMarshal . . . . .	3828
6.823.3.2 looseUnmarshal . . . . .	3828
6.823.3.3 tightMarshal1 . . . . .	3829
6.823.3.4 tightMarshal2 . . . . .	3829
6.823.3.5 tightUnmarshal . . . . .	3830
6.824activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference . . . . .	3831
6.824.1 Detailed Description . . . . .	3831
6.824.2 Constructor & Destructor Documentation . . . . .	3832
6.824.2.1 TransactionIdMarshaller . . . . .	3832
6.824.2.2 ~TransactionIdMarshaller . . . . .	3832
6.824.3 Member Function Documentation . . . . .	3832
6.824.3.1 looseMarshal . . . . .	3832
6.824.3.2 looseUnmarshal . . . . .	3832
6.824.3.3 tightMarshal1 . . . . .	3833
6.824.3.4 tightMarshal2 . . . . .	3833
6.824.3.5 tightUnmarshal . . . . .	3834
6.825activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference . . . . .	3835
6.825.1 Detailed Description . . . . .	3835
6.825.2 Constructor & Destructor Documentation . . . . .	3836
6.825.2.1 TransactionIdMarshaller . . . . .	3836
6.825.2.2 ~TransactionIdMarshaller . . . . .	3836
6.825.3 Member Function Documentation . . . . .	3836
6.825.3.1 looseMarshal . . . . .	3836
6.825.3.2 looseUnmarshal . . . . .	3836
6.825.3.3 tightMarshal1 . . . . .	3837
6.825.3.4 tightMarshal2 . . . . .	3837
6.825.3.5 tightUnmarshal . . . . .	3838
6.826activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Reference . . . . .	3839
6.826.1 Detailed Description . . . . .	3839
6.826.2 Constructor & Destructor Documentation . . . . .	3840
6.826.2.1 TransactionIdMarshaller . . . . .	3840
6.826.2.2 ~TransactionIdMarshaller . . . . .	3840
6.826.3 Member Function Documentation . . . . .	3840

6.826.3.1 looseMarshal . . . . .	3840
6.826.3.2 looseUnmarshal . . . . .	3840
6.826.3.3 tightMarshal1 . . . . .	3841
6.826.3.4 tightMarshal2 . . . . .	3841
6.826.3.5 tightUnmarshal . . . . .	3842
6.827activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller Class Reference . . . . .	3843
6.827.1 Detailed Description . . . . .	3843
6.827.2 Constructor & Destructor Documentation . . . . .	3844
6.827.2.1 TransactionIdMarshaller . . . . .	3844
6.827.2.2 ~TransactionIdMarshaller . . . . .	3844
6.827.3 Member Function Documentation . . . . .	3844
6.827.3.1 looseMarshal . . . . .	3844
6.827.3.2 looseUnmarshal . . . . .	3844
6.827.3.3 tightMarshal1 . . . . .	3845
6.827.3.4 tightMarshal2 . . . . .	3845
6.827.3.5 tightUnmarshal . . . . .	3846
6.828activemq::commands::TransactionInfo Class Reference . . . . .	3847
6.828.1 Constructor & Destructor Documentation . . . . .	3848
6.828.1.1 TransactionInfo . . . . .	3848
6.828.1.2 ~TransactionInfo . . . . .	3848
6.828.2 Member Function Documentation . . . . .	3848
6.828.2.1 cloneDataStructure . . . . .	3848
6.828.2.2 copyDataStructure . . . . .	3848
6.828.2.3 equals . . . . .	3848
6.828.2.4 getConnectionId . . . . .	3849
6.828.2.5 getConnectionId . . . . .	3849
6.828.2.6 getDataStructureType . . . . .	3849
6.828.2.7 getTransactionId . . . . .	3849
6.828.2.8 getTransactionId . . . . .	3849
6.828.2.9 getType . . . . .	3849
6.828.2.10sTransactionInfo . . . . .	3849
6.828.2.11setConnectionId . . . . .	3849
6.828.2.12setTransactionId . . . . .	3849
6.828.2.13setType . . . . .	3849
6.828.2.14toString . . . . .	3849
6.828.2.15visit . . . . .	3850

6.828.3 Field Documentation . . . . .	3850
6.828.3.1 connectionId . . . . .	3850
6.828.3.2 ID_TRANSACTIONINFO . . . . .	3850
6.828.3.3 transactionId . . . . .	3850
6.828.3.4 type . . . . .	3850
6.829activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference . . . . .	3851
6.829.1 Detailed Description . . . . .	3851
6.829.2 Constructor & Destructor Documentation . . . . .	3852
6.829.2.1 TransactionInfoMarshaller . . . . .	3852
6.829.2.2 ~TransactionInfoMarshaller . . . . .	3852
6.829.3 Member Function Documentation . . . . .	3852
6.829.3.1 createObject . . . . .	3852
6.829.3.2 getDataStructureType . . . . .	3852
6.829.3.3 looseMarshal . . . . .	3852
6.829.3.4 looseUnmarshal . . . . .	3853
6.829.3.5 tightMarshal1 . . . . .	3853
6.829.3.6 tightMarshal2 . . . . .	3853
6.829.3.7 tightUnmarshal . . . . .	3854
6.830activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference . . . . .	3855
6.830.1 Detailed Description . . . . .	3855
6.830.2 Constructor & Destructor Documentation . . . . .	3856
6.830.2.1 TransactionInfoMarshaller . . . . .	3856
6.830.2.2 ~TransactionInfoMarshaller . . . . .	3856
6.830.3 Member Function Documentation . . . . .	3856
6.830.3.1 createObject . . . . .	3856
6.830.3.2 getDataStructureType . . . . .	3856
6.830.3.3 looseMarshal . . . . .	3856
6.830.3.4 looseUnmarshal . . . . .	3857
6.830.3.5 tightMarshal1 . . . . .	3857
6.830.3.6 tightMarshal2 . . . . .	3857
6.830.3.7 tightUnmarshal . . . . .	3858
6.831activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference . . . . .	3859
6.831.1 Detailed Description . . . . .	3859
6.831.2 Constructor & Destructor Documentation . . . . .	3860



6.831.2.1 TransactionInfoMarshaller . . . . .	3860
6.831.2.2 ~TransactionInfoMarshaller . . . . .	3860
6.831.3 Member Function Documentation . . . . .	3860
6.831.3.1 createObject . . . . .	3860
6.831.3.2 getDataStructureType . . . . .	3860
6.831.3.3 looseMarshal . . . . .	3860
6.831.3.4 looseUnmarshal . . . . .	3861
6.831.3.5 tightMarshal1 . . . . .	3861
6.831.3.6 tightMarshal2 . . . . .	3861
6.831.3.7 tightUnmarshal . . . . .	3862
6.832activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference . . . . .	3863
6.832.1 Detailed Description . . . . .	3863
6.832.2 Constructor & Destructor Documentation . . . . .	3864
6.832.2.1 TransactionInfoMarshaller . . . . .	3864
6.832.2.2 ~TransactionInfoMarshaller . . . . .	3864
6.832.3 Member Function Documentation . . . . .	3864
6.832.3.1 createObject . . . . .	3864
6.832.3.2 getDataStructureType . . . . .	3864
6.832.3.3 looseMarshal . . . . .	3864
6.832.3.4 looseUnmarshal . . . . .	3865
6.832.3.5 tightMarshal1 . . . . .	3865
6.832.3.6 tightMarshal2 . . . . .	3865
6.832.3.7 tightUnmarshal . . . . .	3866
6.833activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference . . . . .	3867
6.833.1 Detailed Description . . . . .	3867
6.833.2 Constructor & Destructor Documentation . . . . .	3868
6.833.2.1 TransactionInfoMarshaller . . . . .	3868
6.833.2.2 ~TransactionInfoMarshaller . . . . .	3868
6.833.3 Member Function Documentation . . . . .	3868
6.833.3.1 createObject . . . . .	3868
6.833.3.2 getDataStructureType . . . . .	3868
6.833.3.3 looseMarshal . . . . .	3868
6.833.3.4 looseUnmarshal . . . . .	3869
6.833.3.5 tightMarshal1 . . . . .	3869
6.833.3.6 tightMarshal2 . . . . .	3869

6.833.3.7 tightUnmarshal . . . . .	3870
6.834activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference . . . . .	3871
6.834.1 Detailed Description . . . . .	3871
6.834.2 Constructor & Destructor Documentation . . . . .	3872
6.834.2.1 TransactionInfoMarshaller . . . . .	3872
6.834.2.2 ~TransactionInfoMarshaller . . . . .	3872
6.834.3 Member Function Documentation . . . . .	3872
6.834.3.1 createObject . . . . .	3872
6.834.3.2 getDataStructureType . . . . .	3872
6.834.3.3 looseMarshal . . . . .	3872
6.834.3.4 looseUnmarshal . . . . .	3873
6.834.3.5 tightMarshal1 . . . . .	3873
6.834.3.6 tightMarshal2 . . . . .	3873
6.834.3.7 tightUnmarshal . . . . .	3874
6.835activemq::state::TransactionState Class Reference . . . . .	3875
6.835.1 Constructor & Destructor Documentation . . . . .	3876
6.835.1.1 TransactionState . . . . .	3876
6.835.1.2 ~TransactionState . . . . .	3876
6.835.2 Member Function Documentation . . . . .	3876
6.835.2.1 addCommand . . . . .	3876
6.835.2.2 addProducerState . . . . .	3876
6.835.2.3 checkShutdown . . . . .	3876
6.835.2.4 getCommands . . . . .	3876
6.835.2.5 getId . . . . .	3876
6.835.2.6 getPreparedResult . . . . .	3876
6.835.2.7 getProducerStates . . . . .	3876
6.835.2.8 isPrepared . . . . .	3876
6.835.2.9 setPrepared . . . . .	3876
6.835.2.10setPreparedResult . . . . .	3876
6.835.2.11shutdown . . . . .	3876
6.835.2.12toString . . . . .	3876
6.836decaf::internal::util::concurrent::Transferer< E > Class Template Reference . . . . .	3877
6.836.1 Detailed Description . . . . .	3877
6.837decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference . . . . .	3878
6.837.1 Detailed Description . . . . .	3878
6.837.2 Constructor & Destructor Documentation . . . . .	3878

6.837.2.1	TransferQueue . . . . .	3878
6.837.2.2	~TransferQueue . . . . .	3879
6.837.3	Member Function Documentation . . . . .	3879
6.837.3.1	transfer . . . . .	3879
6.837.3.2	transfer . . . . .	3879
6.838	decaf::internal::util::concurrent::TransferStack< E > Class Template Reference . . . . .	3881
6.838.1	Constructor & Destructor Documentation . . . . .	3881
6.838.1.1	TransferStack . . . . .	3881
6.838.1.2	~TransferStack . . . . .	3881
6.838.2	Member Function Documentation . . . . .	3881
6.838.2.1	transfer . . . . .	3881
6.838.2.2	transfer . . . . .	3882
6.839	activemq::transport::Transport Class Reference . . . . .	3883
6.839.1	Detailed Description . . . . .	3884
6.839.2	Constructor & Destructor Documentation . . . . .	3884
6.839.2.1	~Transport . . . . .	3884
6.839.3	Member Function Documentation . . . . .	3884
6.839.3.1	getRemoteAddress . . . . .	3884
6.839.3.2	getTransportListener . . . . .	3884
6.839.3.3	isClosed . . . . .	3885
6.839.3.4	isConnected . . . . .	3885
6.839.3.5	isFaultTolerant . . . . .	3885
6.839.3.6	narrow . . . . .	3885
6.839.3.7	oneway . . . . .	3886
6.839.3.8	reconnect . . . . .	3886
6.839.3.9	request . . . . .	3886
6.839.3.10	request . . . . .	3887
6.839.3.11	setTransportListener . . . . .	3887
6.839.3.12	setWireFormat . . . . .	3888
6.839.3.13	start . . . . .	3888
6.839.3.14	stop . . . . .	3888
6.840	activemq::transport::TransportFactory Class Reference . . . . .	3889
6.840.1	Detailed Description . . . . .	3889
6.840.2	Constructor & Destructor Documentation . . . . .	3889
6.840.2.1	~TransportFactory . . . . .	3889
6.840.3	Member Function Documentation . . . . .	3889

6.840.3.1 create . . . . .	3889
6.840.3.2 createComposite . . . . .	3890
6.841activemq::transport::TransportFilter Class Reference . . . . .	3891
6.841.1 Detailed Description . . . . .	3893
6.841.2 Constructor & Destructor Documentation . . . . .	3893
6.841.2.1 TransportFilter . . . . .	3893
6.841.2.2 ~TransportFilter . . . . .	3893
6.841.3 Member Function Documentation . . . . .	3893
6.841.3.1 close . . . . .	3893
6.841.3.2 fire . . . . .	3893
6.841.3.3 fire . . . . .	3894
6.841.3.4 getRemoteAddress . . . . .	3894
6.841.3.5 getTransportListener . . . . .	3894
6.841.3.6 isClosed . . . . .	3894
6.841.3.7 isConnected . . . . .	3894
6.841.3.8 isFaultTolerant . . . . .	3895
6.841.3.9 narrow . . . . .	3895
6.841.3.10onCommand . . . . .	3895
6.841.3.11oneway . . . . .	3895
6.841.3.12onException . . . . .	3896
6.841.3.13reconnect . . . . .	3896
6.841.3.14request . . . . .	3896
6.841.3.15request . . . . .	3897
6.841.3.16setTransportListener . . . . .	3897
6.841.3.17setWireFormat . . . . .	3897
6.841.3.18start . . . . .	3898
6.841.3.19stop . . . . .	3898
6.841.3.20transportInterrupted . . . . .	3898
6.841.3.21transportResumed . . . . .	3898
6.841.4 Field Documentation . . . . .	3898
6.841.4.1 listener . . . . .	3898
6.841.4.2 next . . . . .	3898
6.842activemq::transport::TransportListener Class Reference . . . . .	3900
6.842.1 Detailed Description . . . . .	3900
6.842.2 Constructor & Destructor Documentation . . . . .	3900
6.842.2.1 ~TransportListener . . . . .	3900

6.842.3 Member Function Documentation . . . . .	3900
6.842.3.1 onCommand . . . . .	3900
6.842.3.2 onException . . . . .	3901
6.842.3.3 transportInterrupted . . . . .	3901
6.842.3.4 transportResumed . . . . .	3901
6.843activemq::transport::TransportRegistry Class Reference . . . . .	3902
6.843.1 Detailed Description . . . . .	3902
6.843.2 Constructor & Destructor Documentation . . . . .	3903
6.843.2.1 ~TransportRegistry . . . . .	3903
6.843.3 Member Function Documentation . . . . .	3903
6.843.3.1 findFactory . . . . .	3903
6.843.3.2 getInstance . . . . .	3903
6.843.3.3 getTransportNames . . . . .	3903
6.843.3.4 registerFactory . . . . .	3903
6.843.3.5 unregisterFactory . . . . .	3904
6.844tree_desc_s Struct Reference . . . . .	3905
6.844.1 Field Documentation . . . . .	3905
6.844.1.1 dyn_tree . . . . .	3905
6.844.1.2 max_code . . . . .	3905
6.844.1.3 stat_desc . . . . .	3905
6.845decaf::lang::Thread::UncaughtExceptionHandler Class Reference . . . . .	3906
6.845.1 Detailed Description . . . . .	3906
6.845.2 Constructor & Destructor Documentation . . . . .	3906
6.845.2.1 ~UncaughtExceptionHandler . . . . .	3906
6.845.3 Member Function Documentation . . . . .	3906
6.845.3.1 uncaughtException . . . . .	3906
6.846decaf::net::UnknownHostException Class Reference . . . . .	3907
6.846.1 Constructor & Destructor Documentation . . . . .	3907
6.846.1.1 UnknownHostException . . . . .	3907
6.846.1.2 UnknownHostException . . . . .	3907
6.846.1.3 UnknownHostException . . . . .	3908
6.846.1.4 UnknownHostException . . . . .	3908
6.846.1.5 UnknownHostException . . . . .	3908
6.846.1.6 UnknownHostException . . . . .	3908
6.846.1.7 ~UnknownHostException . . . . .	3909
6.846.2 Member Function Documentation . . . . .	3909

6.846.2.1 clone . . . . .	3909
6.847decaf::net::UnknownServiceException Class Reference . . . . .	3910
6.847.1 Constructor & Destructor Documentation . . . . .	3910
6.847.1.1 UnknownServiceException . . . . .	3910
6.847.1.2 UnknownServiceException . . . . .	3910
6.847.1.3 UnknownServiceException . . . . .	3911
6.847.1.4 UnknownServiceException . . . . .	3911
6.847.1.5 UnknownServiceException . . . . .	3911
6.847.1.6 UnknownServiceException . . . . .	3911
6.847.1.7 ~UnknownServiceException . . . . .	3912
6.847.2 Member Function Documentation . . . . .	3912
6.847.2.1 clone . . . . .	3912
6.848decaf::io::UnsupportedEncodingException Class Reference . . . . .	3913
6.848.1 Detailed Description . . . . .	3913
6.848.2 Constructor & Destructor Documentation . . . . .	3913
6.848.2.1 UnsupportedEncodingException . . . . .	3913
6.848.2.2 UnsupportedEncodingException . . . . .	3914
6.848.2.3 UnsupportedEncodingException . . . . .	3914
6.848.2.4 UnsupportedEncodingException . . . . .	3914
6.848.2.5 UnsupportedEncodingException . . . . .	3914
6.848.2.6 UnsupportedEncodingException . . . . .	3914
6.848.2.7 ~UnsupportedEncodingException . . . . .	3915
6.848.3 Member Function Documentation . . . . .	3915
6.848.3.1 clone . . . . .	3915
6.849decaf::lang::exceptions::UnsupportedOperationException Class Reference . . . . .	3916
6.849.1 Constructor & Destructor Documentation . . . . .	3916
6.849.1.1 UnsupportedOperationException . . . . .	3916
6.849.1.2 UnsupportedOperationException . . . . .	3916
6.849.1.3 UnsupportedOperationException . . . . .	3917
6.849.1.4 UnsupportedOperationException . . . . .	3917
6.849.1.5 UnsupportedOperationException . . . . .	3917
6.849.1.6 UnsupportedOperationException . . . . .	3917
6.849.1.7 ~UnsupportedOperationException . . . . .	3918
6.849.2 Member Function Documentation . . . . .	3918
6.849.2.1 clone . . . . .	3918
6.850cms::UnsupportedOperationException Class Reference . . . . .	3919

6.850.1 Detailed Description . . . . .	3919
6.850.2 Constructor & Destructor Documentation . . . . .	3919
6.850.2.1 UnsupportedOperationException . . . . .	3919
6.850.2.2 UnsupportedOperationException . . . . .	3919
6.850.2.3 UnsupportedOperationException . . . . .	3919
6.850.2.4 UnsupportedOperationException . . . . .	3919
6.850.2.5 ~UnsupportedOperationException . . . . .	3919
6.851decaf::net::URI Class Reference . . . . .	3921
6.851.1 Detailed Description . . . . .	3923
6.851.2 Constructor & Destructor Documentation . . . . .	3923
6.851.2.1 URI . . . . .	3923
6.851.2.2 URI . . . . .	3923
6.851.2.3 URI . . . . .	3923
6.851.2.4 URI . . . . .	3923
6.851.2.5 URI . . . . .	3924
6.851.2.6 URI . . . . .	3924
6.851.2.7 URI . . . . .	3924
6.851.2.8 ~URI . . . . .	3925
6.851.3 Member Function Documentation . . . . .	3925
6.851.3.1 compareTo . . . . .	3925
6.851.3.2 create . . . . .	3925
6.851.3.3 equals . . . . .	3925
6.851.3.4 getAuthority . . . . .	3925
6.851.3.5 getFragment . . . . .	3925
6.851.3.6 getHost . . . . .	3926
6.851.3.7 getPath . . . . .	3926
6.851.3.8 getPort . . . . .	3926
6.851.3.9 getQuery . . . . .	3926
6.851.3.10getRawAuthority . . . . .	3926
6.851.3.11getRawFragment . . . . .	3926
6.851.3.12getRawPath . . . . .	3926
6.851.3.13getRawQuery . . . . .	3927
6.851.3.14getRawSchemeSpecificPart . . . . .	3927
6.851.3.15getRawUserInfo . . . . .	3927
6.851.3.16getScheme . . . . .	3927
6.851.3.17getSchemeSpecificPart . . . . .	3927

6.851.3.18	getUserInfo . . . . .	3928
6.851.3.19	sAbsolute . . . . .	3928
6.851.3.20	sOpaque . . . . .	3928
6.851.3.21	normalize . . . . .	3928
6.851.3.22	operator< . . . . .	3928
6.851.3.23	operator== . . . . .	3929
6.851.3.24	parseServerAuthority . . . . .	3929
6.851.3.25	relativize . . . . .	3929
6.851.3.26	resolve . . . . .	3930
6.851.3.27	resolve . . . . .	3930
6.851.3.28	oString . . . . .	3931
6.851.3.29	oURL . . . . .	3931
6.852	decaf::internal::net::URIEncoderDecoder Class Reference . . . . .	3932
6.852.1	Constructor & Destructor Documentation . . . . .	3932
6.852.1.1	URIEncoderDecoder . . . . .	3932
6.852.1.2	~URIEncoderDecoder . . . . .	3932
6.852.2	Member Function Documentation . . . . .	3932
6.852.2.1	decode . . . . .	3932
6.852.2.2	encodeOthers . . . . .	3933
6.852.2.3	quoteIllegal . . . . .	3933
6.852.2.4	validate . . . . .	3933
6.852.2.5	validateSimple . . . . .	3934
6.853	decaf::internal::net::URIHelper Class Reference . . . . .	3935
6.853.1	Detailed Description . . . . .	3936
6.853.2	Constructor & Destructor Documentation . . . . .	3936
6.853.2.1	URIHelper . . . . .	3936
6.853.2.2	URIHelper . . . . .	3936
6.853.2.3	~URIHelper . . . . .	3937
6.853.3	Member Function Documentation . . . . .	3937
6.853.3.1	isValidDomainName . . . . .	3937
6.853.3.2	isValidHexChar . . . . .	3937
6.853.3.3	isValidHost . . . . .	3937
6.853.3.4	isValidIP4Word . . . . .	3937
6.853.3.5	isValidIP6Address . . . . .	3938
6.853.3.6	isValidIPv4Address . . . . .	3938
6.853.3.7	parseAuthority . . . . .	3938



6.853.3.8	parseURI . . . . .	3939
6.853.3.9	validateAuthority . . . . .	3939
6.853.3.10	validateFragment . . . . .	3939
6.853.3.11	validatePath . . . . .	3940
6.853.3.12	validateQuery . . . . .	3940
6.853.3.13	validateScheme . . . . .	3940
6.853.3.14	validateSsp . . . . .	3941
6.853.3.15	validateUserinfo . . . . .	3941
6.854	activemq::transport::failover::URIPool Class Reference . . . . .	3942
6.854.1	Constructor & Destructor Documentation . . . . .	3942
6.854.1.1	URIPool . . . . .	3942
6.854.1.2	URIPool . . . . .	3942
6.854.1.3	~URIPool . . . . .	3943
6.854.2	Member Function Documentation . . . . .	3943
6.854.2.1	addURI . . . . .	3943
6.854.2.2	addURIs . . . . .	3943
6.854.2.3	getURI . . . . .	3943
6.854.2.4	isRandomize . . . . .	3943
6.854.2.5	removeURI . . . . .	3943
6.854.2.6	setRandomize . . . . .	3944
6.855	activemq::util::URISupport Class Reference . . . . .	3945
6.855.1	Member Function Documentation . . . . .	3945
6.855.1.1	createQueryString . . . . .	3945
6.855.1.2	parseComposite . . . . .	3946
6.855.1.3	parseQuery . . . . .	3946
6.855.1.4	parseQuery . . . . .	3946
6.855.1.5	parseURL . . . . .	3947
6.856	decaf::net::URISyntaxException Class Reference . . . . .	3948
6.856.1	Constructor & Destructor Documentation . . . . .	3948
6.856.1.1	URISyntaxException . . . . .	3948
6.856.1.2	URISyntaxException . . . . .	3949
6.856.1.3	URISyntaxException . . . . .	3949
6.856.1.4	URISyntaxException . . . . .	3949
6.856.1.5	URISyntaxException . . . . .	3949
6.856.1.6	URISyntaxException . . . . .	3949
6.856.1.7	URISyntaxException . . . . .	3950

6.856.1.8 URISyntaxException . . . . .	3950
6.856.1.9 ~URISyntaxException . . . . .	3950
6.856.2 Member Function Documentation . . . . .	3950
6.856.2.1 clone . . . . .	3950
6.856.2.2 getIndex . . . . .	3951
6.856.2.3 getInput . . . . .	3951
6.856.2.4 getReason . . . . .	3951
6.857decaf::internal::net::URIType Class Reference . . . . .	3952
6.857.1 Detailed Description . . . . .	3954
6.857.2 Constructor & Destructor Documentation . . . . .	3954
6.857.2.1 URIType . . . . .	3954
6.857.2.2 URIType . . . . .	3954
6.857.2.3 ~URIType . . . . .	3954
6.857.3 Member Function Documentation . . . . .	3954
6.857.3.1 getAuthority . . . . .	3954
6.857.3.2 getFragment . . . . .	3954
6.857.3.3 getHost . . . . .	3954
6.857.3.4 getPath . . . . .	3954
6.857.3.5 getPort . . . . .	3955
6.857.3.6 getQuery . . . . .	3955
6.857.3.7 getScheme . . . . .	3955
6.857.3.8 getSchemeSpecificPart . . . . .	3955
6.857.3.9 getSource . . . . .	3955
6.857.3.10getUserInfo . . . . .	3955
6.857.3.11isAbsolute . . . . .	3956
6.857.3.12sOpaque . . . . .	3956
6.857.3.13sServerAuthority . . . . .	3956
6.857.3.14sValid . . . . .	3956
6.857.3.15setAbsolute . . . . .	3956
6.857.3.16setAuthority . . . . .	3956
6.857.3.17setFragment . . . . .	3957
6.857.3.18setHost . . . . .	3957
6.857.3.19setOpaque . . . . .	3957
6.857.3.20setPath . . . . .	3957
6.857.3.21setPort . . . . .	3957
6.857.3.22setQuery . . . . .	3957

6.857.3.23	setScheme . . . . .	3958
6.857.3.24	setSchemeSpecificPart . . . . .	3958
6.857.3.25	setServerAuthority . . . . .	3958
6.857.3.26	setSource . . . . .	3958
6.857.3.27	setUserInfo . . . . .	3958
6.857.3.28	setValid . . . . .	3958
6.858	decaf::net::URL Class Reference . . . . .	3960
6.858.1	Detailed Description . . . . .	3960
6.858.2	Constructor & Destructor Documentation . . . . .	3961
6.858.2.1	URL . . . . .	3961
6.858.2.2	URL . . . . .	3961
6.858.2.3	~URL . . . . .	3961
6.859	decaf::net::URLDecoder Class Reference . . . . .	3962
6.859.1	Constructor & Destructor Documentation . . . . .	3962
6.859.1.1	~URLDecoder . . . . .	3962
6.859.2	Member Function Documentation . . . . .	3962
6.859.2.1	decode . . . . .	3962
6.860	decaf::net::URLEncoder Class Reference . . . . .	3963
6.860.1	Constructor & Destructor Documentation . . . . .	3963
6.860.1.1	~URLEncoder . . . . .	3963
6.860.2	Member Function Documentation . . . . .	3963
6.860.2.1	encode . . . . .	3963
6.861	activemq::util::Usage Class Reference . . . . .	3964
6.861.1	Constructor & Destructor Documentation . . . . .	3964
6.861.1.1	~Usage . . . . .	3964
6.861.2	Member Function Documentation . . . . .	3964
6.861.2.1	decreaseUsage . . . . .	3964
6.861.2.2	enqueueUsage . . . . .	3964
6.861.2.3	increaseUsage . . . . .	3965
6.861.2.4	isFull . . . . .	3965
6.861.2.5	waitForSpace . . . . .	3965
6.861.2.6	waitForSpace . . . . .	3965
6.862	decaf::io::UTFDataFormatException Class Reference . . . . .	3966
6.862.1	Detailed Description . . . . .	3966
6.862.2	Constructor & Destructor Documentation . . . . .	3966
6.862.2.1	UTFDataFormatException . . . . .	3966

6.862.2.2 UTFDataFormatException . . . . .	3967
6.862.2.3 UTFDataFormatException . . . . .	3967
6.862.2.4 UTFDataFormatException . . . . .	3967
6.862.2.5 UTFDataFormatException . . . . .	3967
6.862.2.6 UTFDataFormatException . . . . .	3967
6.862.2.7 ~UTFDataFormatException . . . . .	3968
6.862.3 Member Function Documentation . . . . .	3968
6.862.3.1 clone . . . . .	3968
6.863decaf::util::UUID Class Reference . . . . .	3969
6.863.1 Detailed Description . . . . .	3970
6.863.2 Constructor & Destructor Documentation . . . . .	3970
6.863.2.1 UUID . . . . .	3970
6.863.2.2 ~UUID . . . . .	3971
6.863.3 Member Function Documentation . . . . .	3971
6.863.3.1 clockSequence . . . . .	3971
6.863.3.2 compareTo . . . . .	3971
6.863.3.3 equals . . . . .	3971
6.863.3.4 fromString . . . . .	3972
6.863.3.5 getLeastSignificantBits . . . . .	3972
6.863.3.6 getMostSignificantBits . . . . .	3972
6.863.3.7 nameUUIDFromBytes . . . . .	3972
6.863.3.8 nameUUIDFromBytes . . . . .	3972
6.863.3.9 node . . . . .	3973
6.863.3.10operator< . . . . .	3973
6.863.3.11operator== . . . . .	3973
6.863.3.12randomUUID . . . . .	3973
6.863.3.13timestamp . . . . .	3974
6.863.3.14toString . . . . .	3974
6.863.3.15variant . . . . .	3974
6.863.3.16version . . . . .	3974
6.864activemq::wireformat::WireFormat Class Reference . . . . .	3976
6.864.1 Detailed Description . . . . .	3976
6.864.2 Constructor & Destructor Documentation . . . . .	3977
6.864.2.1 ~WireFormat . . . . .	3977
6.864.3 Member Function Documentation . . . . .	3977
6.864.3.1 createNegotiator . . . . .	3977

6.864.3.2	getVersion	3977
6.864.3.3	hasNegotiator	3977
6.864.3.4	inReceive	3978
6.864.3.5	marshal	3978
6.864.3.6	setVersion	3978
6.864.3.7	unmarshal	3979
6.865	activemq::wireformat::WireFormatFactory Class Reference	3980
6.865.1	Detailed Description	3980
6.865.2	Constructor & Destructor Documentation	3980
6.865.2.1	~WireFormatFactory	3980
6.865.3	Member Function Documentation	3980
6.865.3.1	createWireFormat	3980
6.866	activemq::commands::WireFormatInfo Class Reference	3982
6.866.1	Constructor & Destructor Documentation	3984
6.866.1.1	WireFormatInfo	3984
6.866.1.2	~WireFormatInfo	3984
6.866.2	Member Function Documentation	3984
6.866.2.1	afterUnmarshal	3984
6.866.2.2	beforeMarshal	3985
6.866.2.3	cloneDataStructure	3985
6.866.2.4	copyDataStructure	3985
6.866.2.5	equals	3985
6.866.2.6	getCacheSize	3986
6.866.2.7	getDataStructureType	3986
6.866.2.8	getMagic	3986
6.866.2.9	getMarshaledProperties	3986
6.866.2.10	getMaxInactivityDuration	3986
6.866.2.11	getMaxInactivityDurationInitialDelay	3987
6.866.2.12	getProperties	3987
6.866.2.13	getProperties	3987
6.866.2.14	getVersion	3987
6.866.2.15	isCacheEnabled	3987
6.866.2.16	isMarshalAware	3987
6.866.2.17	isSizePrefixDisabled	3988
6.866.2.18	isStackTraceEnabled	3988
6.866.2.19	isTcpNoDelayEnabled	3988

6.866.2.20	isTightEncodingEnabled . . . . .	3988
6.866.2.21	isValid . . . . .	3988
6.866.2.22	isWireFormatInfo . . . . .	3989
6.866.2.23	setCacheEnabled . . . . .	3989
6.866.2.24	setCacheSize . . . . .	3989
6.866.2.25	setMagic . . . . .	3989
6.866.2.26	setMarshaledProperties . . . . .	3989
6.866.2.27	setMaxInactivityDuration . . . . .	3989
6.866.2.28	setMaxInactivityDurationInitialDelay . . . . .	3990
6.866.2.29	setProperties . . . . .	3990
6.866.2.30	setSizePrefixDisabled . . . . .	3990
6.866.2.31	setStackTraceEnabled . . . . .	3990
6.866.2.32	setTcpNoDelayEnabled . . . . .	3990
6.866.2.33	setTightEncodingEnabled . . . . .	3990
6.866.2.34	setVersion . . . . .	3991
6.866.2.35	toString . . . . .	3991
6.866.2.36	visit . . . . .	3991
6.866.3	Field Documentation . . . . .	3991
6.866.3.1	ID_WIREFORMATINFO . . . . .	3991
6.867	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference . . . . .	3992
6.867.1	Detailed Description . . . . .	3992
6.867.2	Constructor & Destructor Documentation . . . . .	3993
6.867.2.1	WireFormatInfoMarshaller . . . . .	3993
6.867.2.2	~WireFormatInfoMarshaller . . . . .	3993
6.867.3	Member Function Documentation . . . . .	3993
6.867.3.1	createObject . . . . .	3993
6.867.3.2	getDataStructureType . . . . .	3993
6.867.3.3	looseMarshal . . . . .	3993
6.867.3.4	looseUnmarshal . . . . .	3994
6.867.3.5	tightMarshal1 . . . . .	3994
6.867.3.6	tightMarshal2 . . . . .	3994
6.867.3.7	tightUnmarshal . . . . .	3995
6.868	activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller Class Reference . . . . .	3996
6.868.1	Detailed Description . . . . .	3996
6.868.2	Constructor & Destructor Documentation . . . . .	3997

6.868.2.1 WireFormatInfoMarshaller . . . . .	3997
6.868.2.2 ~WireFormatInfoMarshaller . . . . .	3997
6.868.3 Member Function Documentation . . . . .	3997
6.868.3.1 createObject . . . . .	3997
6.868.3.2 getDataStructureType . . . . .	3997
6.868.3.3 looseMarshal . . . . .	3997
6.868.3.4 looseUnmarshal . . . . .	3998
6.868.3.5 tightMarshal1 . . . . .	3998
6.868.3.6 tightMarshal2 . . . . .	3998
6.868.3.7 tightUnmarshal . . . . .	3999
6.869activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference . . . . .	4000
6.869.1 Detailed Description . . . . .	4000
6.869.2 Constructor & Destructor Documentation . . . . .	4001
6.869.2.1 WireFormatInfoMarshaller . . . . .	4001
6.869.2.2 ~WireFormatInfoMarshaller . . . . .	4001
6.869.3 Member Function Documentation . . . . .	4001
6.869.3.1 createObject . . . . .	4001
6.869.3.2 getDataStructureType . . . . .	4001
6.869.3.3 looseMarshal . . . . .	4001
6.869.3.4 looseUnmarshal . . . . .	4002
6.869.3.5 tightMarshal1 . . . . .	4002
6.869.3.6 tightMarshal2 . . . . .	4002
6.869.3.7 tightUnmarshal . . . . .	4003
6.870activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference . . . . .	4004
6.870.1 Detailed Description . . . . .	4004
6.870.2 Constructor & Destructor Documentation . . . . .	4005
6.870.2.1 WireFormatInfoMarshaller . . . . .	4005
6.870.2.2 ~WireFormatInfoMarshaller . . . . .	4005
6.870.3 Member Function Documentation . . . . .	4005
6.870.3.1 createObject . . . . .	4005
6.870.3.2 getDataStructureType . . . . .	4005
6.870.3.3 looseMarshal . . . . .	4005
6.870.3.4 looseUnmarshal . . . . .	4006
6.870.3.5 tightMarshal1 . . . . .	4006
6.870.3.6 tightMarshal2 . . . . .	4006

6.870.3.7 tightUnmarshal . . . . .	4007
6.871activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference . . . . .	4008
6.871.1 Detailed Description . . . . .	4008
6.871.2 Constructor & Destructor Documentation . . . . .	4009
6.871.2.1 WireFormatInfoMarshaller . . . . .	4009
6.871.2.2 ~WireFormatInfoMarshaller . . . . .	4009
6.871.3 Member Function Documentation . . . . .	4009
6.871.3.1 createObject . . . . .	4009
6.871.3.2 getDataStructureType . . . . .	4009
6.871.3.3 looseMarshal . . . . .	4009
6.871.3.4 looseUnmarshal . . . . .	4010
6.871.3.5 tightMarshal1 . . . . .	4010
6.871.3.6 tightMarshal2 . . . . .	4010
6.871.3.7 tightUnmarshal . . . . .	4011
6.872activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference . . . . .	4012
6.872.1 Detailed Description . . . . .	4012
6.872.2 Constructor & Destructor Documentation . . . . .	4013
6.872.2.1 WireFormatInfoMarshaller . . . . .	4013
6.872.2.2 ~WireFormatInfoMarshaller . . . . .	4013
6.872.3 Member Function Documentation . . . . .	4013
6.872.3.1 createObject . . . . .	4013
6.872.3.2 getDataStructureType . . . . .	4013
6.872.3.3 looseMarshal . . . . .	4013
6.872.3.4 looseUnmarshal . . . . .	4014
6.872.3.5 tightMarshal1 . . . . .	4014
6.872.3.6 tightMarshal2 . . . . .	4014
6.872.3.7 tightUnmarshal . . . . .	4015
6.873activemq::wireformat::WireFormatNegotiator Class Reference . . . . .	4016
6.873.1 Detailed Description . . . . .	4016
6.873.2 Constructor & Destructor Documentation . . . . .	4016
6.873.2.1 WireFormatNegotiator . . . . .	4016
6.873.2.2 ~WireFormatNegotiator . . . . .	4016
6.874activemq::wireformat::WireFormatRegistry Class Reference . . . . .	4017
6.874.1 Detailed Description . . . . .	4017
6.874.2 Constructor & Destructor Documentation . . . . .	4018



6.874.2.1 ~WireFormatRegistry . . . . .	4018
6.874.3 Member Function Documentation . . . . .	4018
6.874.3.1 findFactory . . . . .	4018
6.874.3.2 getInstance . . . . .	4018
6.874.3.3 getWireFormatNames . . . . .	4018
6.874.3.4 registerFactory . . . . .	4019
6.874.3.5 unregisterFactory . . . . .	4019
6.875activemq::transport::inactivity::WriteChecker Class Reference . . . . .	4020
6.875.1 Detailed Description . . . . .	4020
6.875.2 Constructor & Destructor Documentation . . . . .	4020
6.875.2.1 WriteChecker . . . . .	4020
6.875.2.2 ~WriteChecker . . . . .	4020
6.875.3 Member Function Documentation . . . . .	4020
6.875.3.1 run . . . . .	4020
6.876decaf::io::Writer Class Reference . . . . .	4021
6.876.1 Constructor & Destructor Documentation . . . . .	4022
6.876.1.1 Writer . . . . .	4022
6.876.1.2 ~Writer . . . . .	4022
6.876.2 Member Function Documentation . . . . .	4022
6.876.2.1 append . . . . .	4022
6.876.2.2 append . . . . .	4023
6.876.2.3 append . . . . .	4023
6.876.2.4 doAppendChar . . . . .	4024
6.876.2.5 doAppendCharSequence . . . . .	4024
6.876.2.6 doAppendCharSequenceStartEnd . . . . .	4024
6.876.2.7 doWriteArray . . . . .	4024
6.876.2.8 doWriteArrayBounded . . . . .	4024
6.876.2.9 doWriteChar . . . . .	4024
6.876.2.10doWriteString . . . . .	4024
6.876.2.11doWriteStringBounded . . . . .	4024
6.876.2.12doWriteVector . . . . .	4024
6.876.2.13write . . . . .	4024
6.876.2.14write . . . . .	4025
6.876.2.15write . . . . .	4025
6.876.2.16write . . . . .	4025
6.876.2.17write . . . . .	4026

6.876.2.18	write . . . . .	4026
6.877	decaf::security::auth::x500::X500Principal Class Reference . . . . .	4027
6.877.1	Constructor & Destructor Documentation . . . . .	4027
6.877.1.1	~X500Principal . . . . .	4027
6.877.2	Member Function Documentation . . . . .	4027
6.877.2.1	getEncoded . . . . .	4027
6.877.2.2	getName . . . . .	4027
6.877.2.3	hashCode . . . . .	4027
6.878	decaf::security::cert::X509Certificate Class Reference . . . . .	4028
6.878.1	Detailed Description . . . . .	4028
6.878.2	Constructor & Destructor Documentation . . . . .	4029
6.878.2.1	~X509Certificate . . . . .	4029
6.878.3	Member Function Documentation . . . . .	4029
6.878.3.1	checkValidity . . . . .	4029
6.878.3.2	checkValidity . . . . .	4029
6.878.3.3	getBasicConstraints . . . . .	4029
6.878.3.4	getIssuerUniqueID . . . . .	4029
6.878.3.5	getIssuerX500Principal . . . . .	4029
6.878.3.6	getKeyUsage . . . . .	4029
6.878.3.7	getNotAfter . . . . .	4029
6.878.3.8	getNotBefore . . . . .	4029
6.878.3.9	getSigAlgName . . . . .	4029
6.878.3.10	getSigAlgOID . . . . .	4029
6.878.3.11	getSigAlgParams . . . . .	4029
6.878.3.12	getSignature . . . . .	4029
6.878.3.13	getSubjectUniqueID . . . . .	4029
6.878.3.14	getSubjectX500Principal . . . . .	4029
6.878.3.15	getTBSCertificate . . . . .	4029
6.878.3.16	getVersion . . . . .	4029
6.879	activemq::commands::XATransactionId Class Reference . . . . .	4031
6.879.1	Member Typedef Documentation . . . . .	4032
6.879.1.1	COMPARATOR . . . . .	4032
6.879.2	Constructor & Destructor Documentation . . . . .	4032
6.879.2.1	XATransactionId . . . . .	4032
6.879.2.2	XATransactionId . . . . .	4032
6.879.2.3	~XATransactionId . . . . .	4032

6.879.3 Member Function Documentation . . . . .	4032
6.879.3.1 cloneDataStructure . . . . .	4032
6.879.3.2 compareTo . . . . .	4032
6.879.3.3 copyDataStructure . . . . .	4032
6.879.3.4 equals . . . . .	4033
6.879.3.5 equals . . . . .	4033
6.879.3.6 getBranchQualifier . . . . .	4033
6.879.3.7 getBranchQualifier . . . . .	4033
6.879.3.8 getDataStructureType . . . . .	4033
6.879.3.9 getFormatId . . . . .	4034
6.879.3.10 getGlobalTransactionId . . . . .	4034
6.879.3.11 getGlobalTransactionId . . . . .	4034
6.879.3.12 operator< . . . . .	4034
6.879.3.13 operator= . . . . .	4034
6.879.3.14 operator== . . . . .	4034
6.879.3.15 setBranchQualifier . . . . .	4034
6.879.3.16 setFormatId . . . . .	4034
6.879.3.17 setGlobalTransactionId . . . . .	4034
6.879.3.18 toString . . . . .	4034
6.879.4 Field Documentation . . . . .	4035
6.879.4.1 branchQualifier . . . . .	4035
6.879.4.2 formatId . . . . .	4035
6.879.4.3 globalTransactionId . . . . .	4035
6.879.4.4 ID_XATRANSACTIONID . . . . .	4035
6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller Class	
Reference . . . . .	4036
6.880.1 Detailed Description . . . . .	4036
6.880.2 Constructor & Destructor Documentation . . . . .	4037
6.880.2.1 XATransactionIdMarshaller . . . . .	4037
6.880.2.2 ~XATransactionIdMarshaller . . . . .	4037
6.880.3 Member Function Documentation . . . . .	4037
6.880.3.1 createObject . . . . .	4037
6.880.3.2 getDataStructureType . . . . .	4037
6.880.3.3 looseMarshal . . . . .	4037
6.880.3.4 looseUnmarshal . . . . .	4038
6.880.3.5 tightMarshal1 . . . . .	4038
6.880.3.6 tightMarshal2 . . . . .	4038

6.880.3.7 tightUnmarshal . . . . .	4039
6.881activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class	
Reference . . . . .	4040
6.881.1 Detailed Description . . . . .	4040
6.881.2 Constructor & Destructor Documentation . . . . .	4041
6.881.2.1 XATransactionIdMarshaller . . . . .	4041
6.881.2.2 ~XATransactionIdMarshaller . . . . .	4041
6.881.3 Member Function Documentation . . . . .	4041
6.881.3.1 createObject . . . . .	4041
6.881.3.2 getDataStructureType . . . . .	4041
6.881.3.3 looseMarshal . . . . .	4041
6.881.3.4 looseUnmarshal . . . . .	4042
6.881.3.5 tightMarshal1 . . . . .	4042
6.881.3.6 tightMarshal2 . . . . .	4042
6.881.3.7 tightUnmarshal . . . . .	4043
6.882activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class	
Reference . . . . .	4044
6.882.1 Detailed Description . . . . .	4044
6.882.2 Constructor & Destructor Documentation . . . . .	4045
6.882.2.1 XATransactionIdMarshaller . . . . .	4045
6.882.2.2 ~XATransactionIdMarshaller . . . . .	4045
6.882.3 Member Function Documentation . . . . .	4045
6.882.3.1 createObject . . . . .	4045
6.882.3.2 getDataStructureType . . . . .	4045
6.882.3.3 looseMarshal . . . . .	4045
6.882.3.4 looseUnmarshal . . . . .	4046
6.882.3.5 tightMarshal1 . . . . .	4046
6.882.3.6 tightMarshal2 . . . . .	4046
6.882.3.7 tightUnmarshal . . . . .	4047
6.883activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class	
Reference . . . . .	4048
6.883.1 Detailed Description . . . . .	4048
6.883.2 Constructor & Destructor Documentation . . . . .	4049
6.883.2.1 XATransactionIdMarshaller . . . . .	4049
6.883.2.2 ~XATransactionIdMarshaller . . . . .	4049
6.883.3 Member Function Documentation . . . . .	4049
6.883.3.1 createObject . . . . .	4049

6.883.3.2	getDataStructureType . . . . .	4049
6.883.3.3	looseMarshal . . . . .	4049
6.883.3.4	looseUnmarshal . . . . .	4050
6.883.3.5	tightMarshal1 . . . . .	4050
6.883.3.6	tightMarshal2 . . . . .	4050
6.883.3.7	tightUnmarshal . . . . .	4051
6.884	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class	
	Reference . . . . .	4052
6.884.1	Detailed Description . . . . .	4052
6.884.2	Constructor & Destructor Documentation . . . . .	4053
6.884.2.1	XATransactionIdMarshaller . . . . .	4053
6.884.2.2	~XATransactionIdMarshaller . . . . .	4053
6.884.3	Member Function Documentation . . . . .	4053
6.884.3.1	createObject . . . . .	4053
6.884.3.2	getDataStructureType . . . . .	4053
6.884.3.3	looseMarshal . . . . .	4053
6.884.3.4	looseUnmarshal . . . . .	4054
6.884.3.5	tightMarshal1 . . . . .	4054
6.884.3.6	tightMarshal2 . . . . .	4054
6.884.3.7	tightUnmarshal . . . . .	4055
6.885	activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class	
	Reference . . . . .	4056
6.885.1	Detailed Description . . . . .	4056
6.885.2	Constructor & Destructor Documentation . . . . .	4057
6.885.2.1	XATransactionIdMarshaller . . . . .	4057
6.885.2.2	~XATransactionIdMarshaller . . . . .	4057
6.885.3	Member Function Documentation . . . . .	4057
6.885.3.1	createObject . . . . .	4057
6.885.3.2	getDataStructureType . . . . .	4057
6.885.3.3	looseMarshal . . . . .	4057
6.885.3.4	looseUnmarshal . . . . .	4058
6.885.3.5	tightMarshal1 . . . . .	4058
6.885.3.6	tightMarshal2 . . . . .	4058
6.885.3.7	tightUnmarshal . . . . .	4059
6.886	decaf::util::logging::XMLFormatter Class Reference . . . . .	4060
6.886.1	Detailed Description . . . . .	4060
6.886.2	Constructor & Destructor Documentation . . . . .	4060

6.886.2.1 XMLFormatter . . . . .	4060
6.886.2.2 ~XMLFormatter . . . . .	4060
6.886.3 Member Function Documentation . . . . .	4060
6.886.3.1 format . . . . .	4060
6.886.3.2 getHead . . . . .	4061
6.886.3.3 getTail . . . . .	4061
6.887z_stream_s Struct Reference . . . . .	4062
6.887.1 Field Documentation . . . . .	4062
6.887.1.1 Adler . . . . .	4062
6.887.1.2 avail_in . . . . .	4062
6.887.1.3 avail_out . . . . .	4062
6.887.1.4 data_type . . . . .	4062
6.887.1.5 msg . . . . .	4062
6.887.1.6 next_in . . . . .	4062
6.887.1.7 next_out . . . . .	4062
6.887.1.8 opaque . . . . .	4062
6.887.1.9 reserved . . . . .	4062
6.887.1.10 state . . . . .	4062
6.887.1.11 total_in . . . . .	4062
6.887.1.12 total_out . . . . .	4062
6.887.1.13 alloc . . . . .	4062
6.887.1.14 free . . . . .	4062
6.888deflate::util::zip::ZipException Class Reference . . . . .	4064
6.888.1 Constructor & Destructor Documentation . . . . .	4064
6.888.1.1 ZipException . . . . .	4064
6.888.1.2 ZipException . . . . .	4064
6.888.1.3 ZipException . . . . .	4065
6.888.1.4 ZipException . . . . .	4065
6.888.1.5 ZipException . . . . .	4065
6.888.1.6 ZipException . . . . .	4065
6.888.1.7 ~ZipException . . . . .	4066
6.888.2 Member Function Documentation . . . . .	4066
6.888.2.1 clone . . . . .	4066
<b>7 File Documentation</b>	<b>4067</b>
7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference . . . . .	4067
7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference . . . . .	4068

7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference . . . . .	4069
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference . . . . .	4070
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference . . . . .	4071
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference . . . . .	4072
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference . . . . .	4073
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference . . . . .	4074
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference . . . . .	4075
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference . . . . .	4076
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference . . . . .	4077
7.12	src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference . . . . .	4078
7.13	src/main/activemq/cmsutil/SessionCallback.h File Reference . . . . .	4079
7.14	src/main/activemq/cmsutil/SessionPool.h File Reference . . . . .	4080
7.15	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference . . . . .	4081
7.16	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference . . . . .	4082
7.17	src/main/activemq/commands/ActiveMQDestination.h File Reference . . . . .	4083
7.18	src/main/activemq/commands/ActiveMQMapMessage.h File Reference . . . . .	4084
7.19	src/main/activemq/commands/ActiveMQMessage.h File Reference . . . . .	4085
7.20	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference . . . . .	4086
7.21	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference . . . . .	4087
7.22	src/main/activemq/commands/ActiveMQQueue.h File Reference . . . . .	4088
7.23	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference . . . . .	4089
7.24	src/main/activemq/commands/ActiveMQTempDestination.h File Reference . . . . .	4090
7.25	src/main/activemq/commands/ActiveMQTempQueue.h File Reference . . . . .	4091
7.26	src/main/activemq/commands/ActiveMQTempTopic.h File Reference . . . . .	4092
7.27	src/main/activemq/commands/ActiveMQTextMessage.h File Reference . . . . .	4093
7.28	src/main/activemq/commands/ActiveMQTopic.h File Reference . . . . .	4094
7.29	src/main/activemq/commands/BaseCommand.h File Reference . . . . .	4095
7.30	src/main/activemq/commands/BaseDataStructure.h File Reference . . . . .	4096
7.31	src/main/activemq/commands/BooleanExpression.h File Reference . . . . .	4097
7.32	src/main/activemq/commands/BrokerError.h File Reference . . . . .	4098
7.33	src/main/activemq/commands/BrokerId.h File Reference . . . . .	4099
7.34	src/main/activemq/commands/BrokerInfo.h File Reference . . . . .	4100
7.35	src/main/activemq/commands/Command.h File Reference . . . . .	4101
7.36	src/main/activemq/commands/ConnectionControl.h File Reference . . . . .	4102
7.37	src/main/activemq/commands/ConnectionError.h File Reference . . . . .	4103
7.38	src/main/activemq/commands/ConnectionId.h File Reference . . . . .	4104

7.39	src/main/activemq/commands/ConnectionInfo.h File Reference . . . . .	4105
7.40	src/main/activemq/commands/ConsumerControl.h File Reference . . . . .	4106
7.41	src/main/activemq/commands/ConsumerId.h File Reference . . . . .	4107
7.42	src/main/activemq/commands/ConsumerInfo.h File Reference . . . . .	4108
7.43	src/main/activemq/commands/ControlCommand.h File Reference . . . . .	4109
7.44	src/main/activemq/commands/DataArrayResponse.h File Reference . . . . .	4110
7.45	src/main/activemq/commands/DataResponse.h File Reference . . . . .	4111
7.46	src/main/activemq/commands/DataStructure.h File Reference . . . . .	4112
7.47	src/main/activemq/commands/DestinationInfo.h File Reference . . . . .	4113
7.48	src/main/activemq/commands/DiscoveryEvent.h File Reference . . . . .	4114
7.49	src/main/activemq/commands/ExceptionResponse.h File Reference . . . . .	4115
7.50	src/main/activemq/commands/FlushCommand.h File Reference . . . . .	4116
7.51	src/main/activemq/commands/IntegerResponse.h File Reference . . . . .	4117
7.52	src/main/activemq/commands/JournalQueueAck.h File Reference . . . . .	4118
7.53	src/main/activemq/commands/JournalTopicAck.h File Reference . . . . .	4119
7.54	src/main/activemq/commands/JournalTrace.h File Reference . . . . .	4120
7.55	src/main/activemq/commands/JournalTransaction.h File Reference . . . . .	4121
7.56	src/main/activemq/commands/KeepAliveInfo.h File Reference . . . . .	4122
7.57	src/main/activemq/commands/LastPartialCommand.h File Reference . . . . .	4123
7.58	src/main/activemq/commands/LocalTransactionId.h File Reference . . . . .	4124
7.59	src/main/activemq/commands/Message.h File Reference . . . . .	4125
7.60	src/main/cms/Message.h File Reference . . . . .	4126
7.61	src/main/activemq/commands/MessageAck.h File Reference . . . . .	4127
7.62	src/main/activemq/commands/MessageDispatch.h File Reference . . . . .	4128
7.63	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . .	4129
7.64	src/main/activemq/commands/MessageId.h File Reference . . . . .	4130
7.65	src/main/activemq/commands/MessagePull.h File Reference . . . . .	4131
7.66	src/main/activemq/commands/NetworkBridgeFilter.h File Reference . . . . .	4132
7.67	src/main/activemq/commands/PartialCommand.h File Reference . . . . .	4133
7.68	src/main/activemq/commands/ProducerAck.h File Reference . . . . .	4134
7.69	src/main/activemq/commands/ProducerId.h File Reference . . . . .	4135
7.70	src/main/activemq/commands/ProducerInfo.h File Reference . . . . .	4136
7.71	src/main/activemq/commands/RemoveInfo.h File Reference . . . . .	4137
7.72	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference . . . . .	4138
7.73	src/main/activemq/commands/ReplayCommand.h File Reference . . . . .	4139
7.74	src/main/activemq/commands/Response.h File Reference . . . . .	4140



7.75	src/main/activemq/commands/SessionId.h File Reference . . . . .	4141
7.76	src/main/activemq/commands/SessionInfo.h File Reference . . . . .	4142
7.77	src/main/activemq/commands/ShutdownInfo.h File Reference . . . . .	4143
7.78	src/main/activemq/commands/SubscriptionInfo.h File Reference . . . . .	4144
7.79	src/main/activemq/commands/TransactionId.h File Reference . . . . .	4145
7.80	src/main/activemq/commands/TransactionInfo.h File Reference . . . . .	4146
7.81	src/main/activemq/commands/WireFormatInfo.h File Reference . . . . .	4147
7.82	src/main/activemq/commands/XATransactionId.h File Reference . . . . .	4148
7.83	src/main/activemq/core/ActiveMQAckHandler.h File Reference . . . . .	4149
7.84	src/main/activemq/core/ActiveMQConnection.h File Reference . . . . .	4150
7.85	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference . . . . .	4151
7.86	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference . . . . .	4152
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference . . . . .	4153
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference . . . . .	4154
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference . . . . .	4155
7.90	src/main/activemq/core/ActiveMQQueueBrowser.h File Reference . . . . .	4156
7.91	src/main/activemq/core/ActiveMQSession.h File Reference . . . . .	4157
7.92	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference . . . . .	4158
7.93	src/main/activemq/core/ActiveMQTransactionContext.h File Reference . . . . .	4159
7.94	src/main/activemq/core/DispatchData.h File Reference . . . . .	4160
7.95	src/main/activemq/core/Dispatcher.h File Reference . . . . .	4161
7.96	src/main/activemq/core/MessageDispatchChannel.h File Reference . . . . .	4162
7.97	src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference . . . . .	4163
7.98	src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference . . . . .	4164
7.99	src/main/activemq/core/PrefetchPolicy.h File Reference . . . . .	4165
7.100	src/main/activemq/core/RedeliveryPolicy.h File Reference . . . . .	4166
7.101	src/main/activemq/core/Synchronization.h File Reference . . . . .	4167
7.102	src/main/activemq/exceptions/ActiveMQException.h File Reference . . . . .	4168
7.103	src/main/activemq/exceptions/BrokerException.h File Reference . . . . .	4169
7.104	src/main/activemq/exceptions/ExceptionDefines.h File Reference . . . . .	4170
7.104.1	Define Documentation . . . . .	4170
7.104.1.1	AMQ_CATCH_EXCEPTION_CONVERT . . . . .	4170
7.104.1.2	AMQ_CATCH_NOTHROW . . . . .	4170
7.104.1.3	AMQ_CATCH_RETHROW . . . . .	4171
7.104.1.4	AMQ_CATCHALL_NOTHROW . . . . .	4171
7.104.1.5	AMQ_CATCHALL_THROW . . . . .	4171

7.105src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference . . . . .	4173
7.105.1 Define Documentation . . . . .	4173
7.105.1.1 DECAF_CATCH_EXCEPTION_CONVERT . . . . .	4173
7.105.1.2 DECAF_CATCH_NOTHROW . . . . .	4173
7.105.1.3 DECAF_CATCH_RETHROW . . . . .	4174
7.105.1.4 DECAF_CATCHALL_NOTHROW . . . . .	4174
7.105.1.5 DECAF_CATCHALL_THROW . . . . .	4174
7.106src/main/activemq/io/LoggingInputStream.h File Reference . . . . .	4175
7.107src/main/activemq/io/LoggingOutputStream.h File Reference . . . . .	4176
7.108src/main/activemq/library/ActiveMQCPP.h File Reference . . . . .	4177
7.109src/main/activemq/state/CommandVisitor.h File Reference . . . . .	4178
7.110src/main/activemq/state/CommandVisitorAdapter.h File Reference . . . . .	4179
7.111src/main/activemq/state/ConnectionState.h File Reference . . . . .	4181
7.112src/main/activemq/state/ConnectionStateTracker.h File Reference . . . . .	4182
7.113src/main/activemq/state/ConsumerState.h File Reference . . . . .	4183
7.114src/main/activemq/state/ProducerState.h File Reference . . . . .	4184
7.115src/main/activemq/state/SessionState.h File Reference . . . . .	4185
7.116src/main/activemq/state/Tracked.h File Reference . . . . .	4186
7.117src/main/activemq/state/TransactionState.h File Reference . . . . .	4187
7.118src/main/activemq/threads/CompositeTask.h File Reference . . . . .	4188
7.119src/main/activemq/threads/CompositeTaskRunner.h File Reference . . . . .	4189
7.120src/main/activemq/threads/DedicatedTaskRunner.h File Reference . . . . .	4190
7.121src/main/activemq/threads/Task.h File Reference . . . . .	4191
7.122src/main/activemq/threads/TaskRunner.h File Reference . . . . .	4192
7.123src/main/activemq/transport/AbstractTransportFactory.h File Reference . . . . .	4193
7.124src/main/activemq/transport/CompositeTransport.h File Reference . . . . .	4194
7.125src/main/activemq/transport/correlator/FutureResponse.h File Reference . . . . .	4195
7.126src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference . . . . .	4196
7.127src/main/activemq/transport/DefaultTransportListener.h File Reference . . . . .	4197
7.128src/main/activemq/transport/failover/BackupTransport.h File Reference . . . . .	4198
7.129src/main/activemq/transport/failover/BackupTransportPool.h File Reference . . . . .	4199
7.130src/main/activemq/transport/failover/CloseTransportsTask.h File Reference . . . . .	4200
7.131src/main/activemq/transport/failover/FailoverTransport.h File Reference . . . . .	4201
7.132src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference . . . . .	4202
7.133src/main/activemq/transport/failover/FailoverTransportListener.h File Reference . . . . .	4203
7.134src/main/activemq/transport/failover/URIPool.h File Reference . . . . .	4204

7.135src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference . . . . .	4205
7.136src/main/activemq/transport/inactivity/ReadChecker.h File Reference . . . . .	4206
7.137src/main/activemq/transport/inactivity/WriteChecker.h File Reference . . . . .	4207
7.138src/main/activemq/transport/IOTransport.h File Reference . . . . .	4208
7.139src/main/activemq/transport/logging/LoggingTransport.h File Reference . . . . .	4209
7.140src/main/activemq/transport/mock/InternalCommandListener.h File Reference . .	4210
7.141src/main/activemq/transport/mock/MockTransport.h File Reference . . . . .	4211
7.142src/main/activemq/transport/mock/MockTransportFactory.h File Reference . . .	4212
7.143src/main/activemq/transport/mock/ResponseBuilder.h File Reference . . . . .	4213
7.144src/main/activemq/transport/tcp/SslTransport.h File Reference . . . . .	4214
7.145src/main/activemq/transport/tcp/SslTransportFactory.h File Reference . . . . .	4215
7.146src/main/activemq/transport/tcp/TcpTransport.h File Reference . . . . .	4216
7.147src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference . . . . .	4217
7.148src/main/activemq/transport/Transport.h File Reference . . . . .	4218
7.149src/main/activemq/transport/TransportFactory.h File Reference . . . . .	4219
7.150src/main/activemq/transport/TransportFilter.h File Reference . . . . .	4220
7.151src/main/activemq/transport/TransportListener.h File Reference . . . . .	4221
7.152src/main/activemq/transport/TransportRegistry.h File Reference . . . . .	4222
7.153src/main/activemq/util/ActiveMQProperties.h File Reference . . . . .	4223
7.154src/main/activemq/util/CMSExceptionSupport.h File Reference . . . . .	4224
7.154.1 Define Documentation . . . . .	4224
7.154.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION . . . . .	4224
7.155src/main/activemq/util/CompositeData.h File Reference . . . . .	4226
7.156src/main/activemq/util/Config.h File Reference . . . . .	4227
7.156.1 Define Documentation . . . . .	4227
7.156.1.1 AMQCPP_API . . . . .	4227
7.157src/main/cms/Config.h File Reference . . . . .	4228
7.157.1 Define Documentation . . . . .	4228
7.157.1.1 CMS_API . . . . .	4228
7.158src/main/decaf/util/Config.h File Reference . . . . .	4229
7.158.1 Define Documentation . . . . .	4229
7.158.1.1 DECAF_API . . . . .	4229
7.158.1.2 DECAF_UNUSED . . . . .	4229
7.159src/main/activemq/util/IdGenerator.h File Reference . . . . .	4230
7.160src/main/activemq/util/LongSequenceGenerator.h File Reference . . . . .	4231
7.161src/main/activemq/util/MarshallingSupport.h File Reference . . . . .	4232

7.162src/main/activemq/util/MemoryUsage.h File Reference . . . . .	4233
7.163src/main/activemq/util/PrimitiveList.h File Reference . . . . .	4234
7.164src/main/activemq/util/PrimitiveMap.h File Reference . . . . .	4235
7.165src/main/activemq/util/PrimitiveValueConverter.h File Reference . . . . .	4236
7.166src/main/activemq/util/PrimitiveValueNode.h File Reference . . . . .	4237
7.167src/main/activemq/util/URISupport.h File Reference . . . . .	4238
7.168src/main/activemq/util/Usage.h File Reference . . . . .	4239
7.169src/main/activemq/wireformat/MarshalAware.h File Reference . . . . .	4240
7.170src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference . . . . .	4241
7.171src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference . . . . .	4242
7.172src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference . . . . .	4243
7.173src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4244
7.174src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4245
7.175src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4246
7.176src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4247
7.177src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4248
7.178src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4249
7.179src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4250
7.180src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4251
7.181src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4252
7.182src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4253
7.183src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4254
7.184src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4255
7.185src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference . . . . .	4256
7.186src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference . . . . .	4257

7.187	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4258
7.188	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4259
7.189	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4260
7.190	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4261
7.191	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4262
7.192	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4263
7.193	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4264
7.194	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4265
7.195	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4266
7.196	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4267
7.197	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4268
7.198	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4269
7.199	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4270
7.200	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4271
7.201	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4272
7.202	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4273
7.203	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4274
7.204	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4275
7.205	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4276
7.206	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4277
7.207	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4278
7.208	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4279

7.209	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4280
7.210	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4281
7.211	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4282
7.212	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4283
7.213	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4284
7.214	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h	
	File Reference . . . . .	4285
7.215	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4286
7.216	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4287
7.217	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4288
7.218	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4289
7.219	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4290
7.220	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h	
	File Reference . . . . .	4291
7.221	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4292
7.222	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4293
7.223	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4294
7.224	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4295
7.225	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4296
7.226	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h	
	File Reference . . . . .	4297
7.227	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4298
7.228	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4299
7.229	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4300
7.230	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4301

7.231	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4302
7.232	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h	
	File Reference . . . . .	4303
7.233	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4304
7.234	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4305
7.235	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4306
7.236	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4307
7.237	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4308
7.238	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h	
	File Reference . . . . .	4309
7.239	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4310
7.240	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4311
7.241	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4312
7.242	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4313
7.243	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4314
7.244	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h	
	File Reference . . . . .	4315
7.245	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4316
7.246	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4317
7.247	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4318
7.248	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4319
7.249	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4320
7.250	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h	
	File Reference . . . . .	4321
7.251	src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	
	File Reference . . . . .	4322
7.252	src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	
	File Reference . . . . .	4323

7.253src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	
File Reference . . . . .	4324
7.254src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	
File Reference . . . . .	4325
7.255src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	
File Reference . . . . .	4326
7.256src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h	
File Reference . . . . .	4327
7.257src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h	File
Reference . . . . .	4328
7.258src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	File
Reference . . . . .	4329
7.259src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	File
Reference . . . . .	4330
7.260src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	File
Reference . . . . .	4331
7.261src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h	File
Reference . . . . .	4332
7.262src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h	File
Reference . . . . .	4333
7.263src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h	File
Reference . . . . .	4334
7.264src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h	File
Reference . . . . .	4335
7.265src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	File
Reference . . . . .	4336
7.266src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h	File
Reference . . . . .	4337
7.267src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h	File
Reference . . . . .	4338
7.268src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h	File
Reference . . . . .	4339
7.269src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
File Reference . . . . .	4340
7.270src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
File Reference . . . . .	4341
7.271src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
File Reference . . . . .	4342
7.272src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
File Reference . . . . .	4343
7.273src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
File Reference . . . . .	4344
7.274src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h	
File Reference . . . . .	4345



7.275	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4346
7.276	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4347
7.277	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4348
7.278	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4349
7.279	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4350
7.280	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h	
	File Reference . . . . .	4351
7.281	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h	
	File Reference . . . . .	4352
7.282	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	
	File Reference . . . . .	4353
7.283	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	
	File Reference . . . . .	4354
7.284	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	
	File Reference . . . . .	4355
7.285	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h	
	File Reference . . . . .	4356
7.286	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h	
	File Reference . . . . .	4357
7.287	src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4358
7.288	src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4359
7.289	src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4360
7.290	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4361
7.291	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4362
7.292	src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h	
	File Reference . . . . .	4363
7.293	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
	File Reference . . . . .	4364
7.294	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h	
	File Reference . . . . .	4365
7.295	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
	File Reference . . . . .	4366
7.296	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
	File Reference . . . . .	4367

7.297	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
	File Reference . . . . .	4368
7.298	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h	
	File Reference . . . . .	4369
7.299	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h	
	File Reference . . . . .	4370
7.300	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h	
	File Reference . . . . .	4371
7.301	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	
	File Reference . . . . .	4372
7.302	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	
	File Reference . . . . .	4373
7.303	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h	
	File Reference . . . . .	4374
7.304	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h	
	File Reference . . . . .	4375
7.305	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4376
7.306	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4377
7.307	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4378
7.308	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4379
7.309	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4380
7.310	src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h	
	File Reference . . . . .	4381
7.311	src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
	File Reference . . . . .	4382
7.312	src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
	File Reference . . . . .	4383
7.313	src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
	File Reference . . . . .	4384
7.314	src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
	File Reference . . . . .	4385
7.315	src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
	File Reference . . . . .	4386
7.316	src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h	
	File Reference . . . . .	4387
7.317	src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4388
7.318	src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4389

7.319	src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4390
7.320	src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4391
7.321	src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4392
7.322	src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h	
	File Reference . . . . .	4393
7.323	src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h	
	File Reference . . . . .	4394
7.324	src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h	
	File Reference . . . . .	4395
7.325	src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h	
	File Reference . . . . .	4396
7.326	src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h	
	File Reference . . . . .	4397
7.327	src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h	
	File Reference . . . . .	4398
7.328	src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h	
	File Reference . . . . .	4399
7.329	src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h	
	File Reference . . . . .	4400
7.330	src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h	
	File Reference . . . . .	4401
7.331	src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h	
	File Reference . . . . .	4402
7.332	src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h	
	File Reference . . . . .	4403
7.333	src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h	
	File Reference . . . . .	4404
7.334	src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h	
	File Reference . . . . .	4405
7.335	src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4406
7.336	src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4407
7.337	src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4408
7.338	src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4409
7.339	src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4410
7.340	src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h	
	File Reference . . . . .	4411

7.341	src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4412
7.342	src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4413
7.343	src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4414
7.344	src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4415
7.345	src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4416
7.346	src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h	
	File Reference . . . . .	4417
7.347	src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	
	File Reference . . . . .	4418
7.348	src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h	
	File Reference . . . . .	4419
7.349	src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	
	File Reference . . . . .	4420
7.350	src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
	File Reference . . . . .	4421
7.351	src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	
	File Reference . . . . .	4422
7.352	src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h	
	File Reference . . . . .	4423
7.353	src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
	File Reference . . . . .	4424
7.354	src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h	
	File Reference . . . . .	4425
7.355	src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
	File Reference . . . . .	4426
7.356	src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
	File Reference . . . . .	4427
7.357	src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
	File Reference . . . . .	4428
7.358	src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h	
	File Reference . . . . .	4429
7.359	src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4430
7.360	src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4431
7.361	src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4432
7.362	src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4433

7.363	src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4434
7.364	src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h	
	File Reference . . . . .	4435
7.365	src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4436
7.366	src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4437
7.367	src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4438
7.368	src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4439
7.369	src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4440
7.370	src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h	
	File Reference . . . . .	4441
7.371	src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
	File Reference . . . . .	4442
7.372	src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
	File Reference . . . . .	4443
7.373	src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
	File Reference . . . . .	4444
7.374	src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
	File Reference . . . . .	4445
7.375	src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
	File Reference . . . . .	4446
7.376	src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h	
	File Reference . . . . .	4447
7.377	src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
	File Reference . . . . .	4448
7.378	src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
	File Reference . . . . .	4449
7.379	src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
	File Reference . . . . .	4450
7.380	src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
	File Reference . . . . .	4451
7.381	src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
	File Reference . . . . .	4452
7.382	src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h	
	File Reference . . . . .	4453
7.383	src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4454
7.384	src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4455

7.385	src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4456
7.386	src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4457
7.387	src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4458
7.388	src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h	
	File Reference . . . . .	4459
7.389	src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4460
7.390	src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4461
7.391	src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4462
7.392	src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4463
7.393	src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4464
7.394	src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h	
	File Reference . . . . .	4465
7.395	src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4466
7.396	src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4467
7.397	src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4468
7.398	src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4469
7.399	src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4470
7.400	src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h	
	File Reference . . . . .	4471
7.401	src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h	File
	Reference . . . . .	4472
7.402	src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h	File
	Reference . . . . .	4473
7.403	src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	File
	Reference . . . . .	4474
7.404	src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	File
	Reference . . . . .	4475
7.405	src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h	File
	Reference . . . . .	4476
7.406	src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h	File
	Reference . . . . .	4477

7.407src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h	
File Reference . . . . .	4478
7.408src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h	
File Reference . . . . .	4479
7.409src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h	
File Reference . . . . .	4480
7.410src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	
File Reference . . . . .	4481
7.411src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h	
File Reference . . . . .	4482
7.412src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h	
File Reference . . . . .	4483
7.413src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h	
File Reference . . . . .	4484
7.414src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h	
File Reference . . . . .	4485
7.415src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h	
File Reference . . . . .	4486
7.416src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
File Reference . . . . .	4487
7.417src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h	
File Reference . . . . .	4488
7.418src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h	
File Reference . . . . .	4489
7.419src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4490
7.420src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4491
7.421src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4492
7.422src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4493
7.423src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4494
7.424src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4495
7.425src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h	File
Reference . . . . .	4496
7.426src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h	File
Reference . . . . .	4497
7.427src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	File
Reference . . . . .	4498
7.428src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	File
Reference . . . . .	4499

7.429	src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h	File	
	Reference		4500
7.430	src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h	File	
	Reference		4501
7.431	src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	File	
	Reference		4502
7.432	src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h	File	
	Reference		4503
7.433	src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	File	
	Reference		4504
7.434	src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	File	
	Reference		4505
7.435	src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	File	
	Reference		4506
7.436	src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h	File	
	Reference		4507
7.437	src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h		
	File Reference		4508
7.438	src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h		
	File Reference		4509
7.439	src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h		
	File Reference		4510
7.440	src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h		
	File Reference		4511
7.441	src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h		
	File Reference		4512
7.442	src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h		
	File Reference		4513
7.443	src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h		
	File Reference		4514
7.444	src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h		
	File Reference		4515
7.445	src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h		
	File Reference		4516
7.446	src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h		
	File Reference		4517
7.447	src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h		
	File Reference		4518
7.448	src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h		
	File Reference		4519
7.449	src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h		
	File Reference		4520
7.450	src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h		
	File Reference		4521



7.451	src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	
	File Reference . . . . .	4522
7.452	src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
	File Reference . . . . .	4523
7.453	src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
	File Reference . . . . .	4524
7.454	src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h	
	File Reference . . . . .	4525
7.455	src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	
	File Reference . . . . .	4526
7.456	src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h	
	File Reference . . . . .	4527
7.457	src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	
	File Reference . . . . .	4528
7.458	src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
	File Reference . . . . .	4529
7.459	src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	
	File Reference . . . . .	4530
7.460	src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h	
	File Reference . . . . .	4531
7.461	src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	
	File Reference . . . . .	4532
7.462	src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h	
	File Reference . . . . .	4533
7.463	src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	
	File Reference . . . . .	4534
7.464	src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	
	File Reference . . . . .	4535
7.465	src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	
	File Reference . . . . .	4536
7.466	src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h	
	File Reference . . . . .	4537
7.467	src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	
	File Reference . . . . .	4538
7.468	src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h	
	File Reference . . . . .	4539
7.469	src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	
	File Reference . . . . .	4540
7.470	src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
	File Reference . . . . .	4541
7.471	src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	
	File Reference . . . . .	4542
7.472	src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h	
	File Reference . . . . .	4543

7.473	src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	
	File Reference . . . . .	4544
7.474	src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h	
	File Reference . . . . .	4545
7.475	src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	
	File Reference . . . . .	4546
7.476	src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
	File Reference . . . . .	4547
7.477	src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h	
	File Reference . . . . .	4548
7.478	src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h	
	File Reference . . . . .	4549
7.479	src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4550
7.480	src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4551
7.481	src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4552
7.482	src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4553
7.483	src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4554
7.484	src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h	
	File Reference . . . . .	4555
7.485	src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h	
	File Reference . . . . .	4556
7.486	src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h	
	File Reference . . . . .	4557
7.487	src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h	
	File Reference . . . . .	4558
7.488	src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h	
	File Reference . . . . .	4559
7.489	src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h	
	File Reference . . . . .	4560
7.490	src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h	
	File Reference . . . . .	4561
7.491	src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h	File
	Reference . . . . .	4562
7.492	src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h	File
	Reference . . . . .	4563
7.493	src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h	File
	Reference . . . . .	4564
7.494	src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h	File
	Reference . . . . .	4565

7.495	src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h	File	
	Reference		4566
7.496	src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h	File	
	Reference		4567
7.497	src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h	File	
	Reference		4568
7.498	src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h	File	
	Reference		4569
7.499	src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h	File	
	Reference		4570
7.500	src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h	File	
	Reference		4571
7.501	src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h	File	
	Reference		4572
7.502	src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h	File	
	Reference		4573
7.503	src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h	File	
	File Reference		4574
7.504	src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h	File	
	File Reference		4575
7.505	src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	File	
	File Reference		4576
7.506	src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h	File	
	File Reference		4577
7.507	src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	File	
	File Reference		4578
7.508	src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h	File	
	File Reference		4579
7.509	src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	File	
	File Reference		4580
7.510	src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h	File	
	File Reference		4581
7.511	src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	File	
	File Reference		4582
7.512	src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h	File	
	File Reference		4583
7.513	src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	File	
	File Reference		4584
7.514	src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h	File	
	File Reference		4585
7.515	src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	File	
	File Reference		4586
7.516	src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h	File	
	File Reference		4587

7.517src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4588
7.518src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4589
7.519src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4590
7.520src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4591
7.521src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	
File Reference . . . . .	4592
7.522src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h	
File Reference . . . . .	4593
7.523src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	
File Reference . . . . .	4594
7.524src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h	
File Reference . . . . .	4595
7.525src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	
File Reference . . . . .	4596
7.526src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h	
File Reference . . . . .	4597
7.527src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	
File Reference . . . . .	4598
7.528src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h	
File Reference . . . . .	4599
7.529src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	
File Reference . . . . .	4600
7.530src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h	
File Reference . . . . .	4601
7.531src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h	
File Reference . . . . .	4602
7.532src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h	
File Reference . . . . .	4603
7.533src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h	
File Reference . . . . .	4604
7.534src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h	
File Reference . . . . .	4605
7.535src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h	
File Reference . . . . .	4606
7.536src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h	
File Reference . . . . .	4607
7.537src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h	
File Reference . . . . .	4608
7.538src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h	
File Reference . . . . .	4609

7.539	src/main/activemq/wireformat/openwire/marshall/v1/XATransactionIdMarshaller.h	
	File Reference . . . . .	4610
7.540	src/main/activemq/wireformat/openwire/marshall/v2/XATransactionIdMarshaller.h	
	File Reference . . . . .	4611
7.541	src/main/activemq/wireformat/openwire/marshall/v3/XATransactionIdMarshaller.h	
	File Reference . . . . .	4612
7.542	src/main/activemq/wireformat/openwire/marshall/v4/XATransactionIdMarshaller.h	
	File Reference . . . . .	4613
7.543	src/main/activemq/wireformat/openwire/marshall/v5/XATransactionIdMarshaller.h	
	File Reference . . . . .	4614
7.544	src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller.h	
	File Reference . . . . .	4615
7.545	src/main/activemq/wireformat/openwire/OpenWireFormat.h	File Reference . . . . .
		4616
7.546	src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h	File Reference
		4617
7.547	src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h	File Ref-
	erence . . . . .	4618
7.548	src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h	File Refer-
	ence . . . . .	4619
7.549	src/main/activemq/wireformat/openwire/utis/BooleanStream.h	File Reference . . . . .
		4620
7.550	src/main/activemq/wireformat/openwire/utis/HexTable.h	File Reference . . . . .
		4621
7.551	src/main/activemq/wireformat/openwire/utis/MessagePropertyInterceptor.h	File
	Reference . . . . .	4622
7.552	src/main/activemq/wireformat/stomp/StompCommandConstants.h	File Reference
		4623
7.553	src/main/activemq/wireformat/stomp/StompFrame.h	File Reference . . . . .
		4624
7.554	src/main/activemq/wireformat/stomp/StompHelper.h	File Reference . . . . .
		4625
7.555	src/main/activemq/wireformat/stomp/StompWireFormat.h	File Reference . . . . .
		4626
7.556	src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	File Reference
		4627
7.557	src/main/activemq/wireformat/WireFormat.h	File Reference . . . . .
		4628
7.558	src/main/activemq/wireformat/WireFormatFactory.h	File Reference . . . . .
		4629
7.559	src/main/activemq/wireformat/WireFormatNegotiator.h	File Reference . . . . .
		4630
7.560	src/main/activemq/wireformat/WireFormatRegistry.h	File Reference . . . . .
		4631
7.561	src/main/cms/BytesMessage.h	File Reference . . . . .
		4632
7.562	src/main/cms/Closeable.h	File Reference . . . . .
		4633
7.563	src/main/decaf/io/Closeable.h	File Reference . . . . .
		4634
7.564	src/main/cms/CMSException.h	File Reference . . . . .
		4635
7.565	src/main/cms/CMSProperties.h	File Reference . . . . .
		4636
7.566	src/main/cms/CMSSecurityException.h	File Reference . . . . .
		4637
7.567	src/main/cms/Connection.h	File Reference . . . . .
		4638
7.568	src/main/cms/ConnectionFactory.h	File Reference . . . . .
		4639

7.569src/main/cms/ConnectionMetaData.h File Reference . . . . .	4640
7.570src/main/cms/DeliveryMode.h File Reference . . . . .	4641
7.571src/main/cms/Destination.h File Reference . . . . .	4642
7.572src/main/cms/ExceptionListener.h File Reference . . . . .	4643
7.573src/main/cms/IllegalStateException.h File Reference . . . . .	4644
7.574src/main/decaf/lang/exceptions/IllegalStateException.h File Reference . . . . .	4645
7.575src/main/cms/InvalidClientIdException.h File Reference . . . . .	4646
7.576src/main/cms/InvalidDestinationException.h File Reference . . . . .	4647
7.577src/main/cms/InvalidSelectorException.h File Reference . . . . .	4648
7.578src/main/cms/MapMessage.h File Reference . . . . .	4649
7.579src/main/cms/MessageConsumer.h File Reference . . . . .	4650
7.580src/main/cms/MessageEnumeration.h File Reference . . . . .	4651
7.581src/main/cms/MessageEOFException.h File Reference . . . . .	4652
7.582src/main/cms/MessageFormatException.h File Reference . . . . .	4653
7.583src/main/cms/MessageListener.h File Reference . . . . .	4654
7.584src/main/cms/MessageNotReadableException.h File Reference . . . . .	4655
7.585src/main/cms/MessageNotWriteableException.h File Reference . . . . .	4656
7.586src/main/cms/MessageProducer.h File Reference . . . . .	4657
7.587src/main/cms/ObjectMessage.h File Reference . . . . .	4658
7.588src/main/cms/Queue.h File Reference . . . . .	4659
7.589src/main/decaf/util/Queue.h File Reference . . . . .	4660
7.590src/main/cms/QueueBrowser.h File Reference . . . . .	4661
7.591src/main/cms/Session.h File Reference . . . . .	4662
7.592src/main/cms/Startable.h File Reference . . . . .	4663
7.593src/main/cms/Stoppable.h File Reference . . . . .	4664
7.594src/main/cms/StreamMessage.h File Reference . . . . .	4665
7.595src/main/cms/TemporaryQueue.h File Reference . . . . .	4666
7.596src/main/cms/TemporaryTopic.h File Reference . . . . .	4667
7.597src/main/cms/TextMessage.h File Reference . . . . .	4668
7.598src/main/cms/Topic.h File Reference . . . . .	4669
7.599src/main/cms/UnsupportedOperationException.h File Reference . . . . .	4670
7.600src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference . . . . .	4671
7.601src/main/decaf/internal/AprPool.h File Reference . . . . .	4672
7.602src/main/decaf/internal/DecafRuntime.h File Reference . . . . .	4673
7.603src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference . . . . .	4674
7.604src/main/decaf/internal/io/StandardInputStream.h File Reference . . . . .	4675

7.605src/main/decaf/internal/io/StandardOutputStream.h File Reference . . . . .	4676
7.606src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference . . . . .	4677
7.607src/main/decaf/internal/net/DefaultSocketFactory.h File Reference . . . . .	4678
7.608src/main/decaf/internal/net/Network.h File Reference . . . . .	4679
7.609src/main/decaf/internal/net/SocketFileDescriptor.h File Reference . . . . .	4680
7.610src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference . . . . .	4681
7.611src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference .	4682
7.612src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference . . . .	4683
7.613src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference . .	4684
7.614src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference . .	4685
7.615src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference .	4686
7.616src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Ref- erence . . . . .	4687
7.617src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference . . . . .	4688
7.618src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference	4689
7.619src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference .	4690
7.620src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Ref- erence . . . . .	4691
7.621src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference . . . . .	4692
7.622src/main/decaf/internal/net/tcp/TcpSocket.h File Reference . . . . .	4693
7.623src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference . . . . .	4694
7.624src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference . . . .	4695
7.625src/main/decaf/internal/net/URIEncoderDecoder.h File Reference . . . . .	4696
7.626src/main/decaf/internal/net/URIHelper.h File Reference . . . . .	4697
7.627src/main/decaf/internal/net/URIType.h File Reference . . . . .	4698
7.628src/main/decaf/internal/nio/BufferFactory.h File Reference . . . . .	4699
7.629src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference . . . . .	4700
7.630src/main/decaf/internal/nio/CharArrayBuffer.h File Reference . . . . .	4701
7.631src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference . . . . .	4702
7.632src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference . . . . .	4703
7.633src/main/decaf/internal/nio/IntArrayBuffer.h File Reference . . . . .	4704
7.634src/main/decaf/internal/nio/LongArrayBuffer.h File Reference . . . . .	4705
7.635src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference . . . . .	4706
7.636src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference . . . .	4707
7.637src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference . .	4708
7.638src/main/decaf/internal/util/ByteArrayAdapter.h File Reference . . . . .	4709

7.639	src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference . . . . .	4710
7.640	src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference . . . . .	4711
7.641	src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference . . .	4712
7.642	src/main/decaf/internal/util/concurrent/Transferer.h File Reference . . . . .	4713
7.643	src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference . . . . .	4714
7.644	src/main/decaf/internal/util/concurrent/TransferStack.h File Reference . . . . .	4715
7.645	src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference .	4716
7.646	src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference	4717
7.647	src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference . . .	4718
7.648	src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference .	4719
7.649	src/main/decaf/internal/util/GenericResource.h File Reference . . . . .	4720
7.650	src/main/decaf/internal/util/HexStringParser.h File Reference . . . . .	4721
7.651	src/main/decaf/internal/util/Resource.h File Reference . . . . .	4722
7.652	src/main/decaf/internal/util/TimerTaskHeap.h File Reference . . . . .	4723
7.653	src/main/decaf/internal/util/zip/crc32.h File Reference . . . . .	4724
7.653.1	Variable Documentation . . . . .	4724
7.653.1.1	crc_table . . . . .	4724
7.654	src/main/decaf/internal/util/zip/deflate.h File Reference . . . . .	4725
7.654.1	Define Documentation . . . . .	4726
7.654.1.1	_tr_tally_dist . . . . .	4726
7.654.1.2	_tr_tally_lit . . . . .	4726
7.654.1.3	BL_CODES . . . . .	4727
7.654.1.4	BUSY_STATE . . . . .	4727
7.654.1.5	Code . . . . .	4727
7.654.1.6	COMMENT_STATE . . . . .	4727
7.654.1.7	d_code . . . . .	4727
7.654.1.8	D_CODES . . . . .	4727
7.654.1.9	Dad . . . . .	4727
7.654.1.10	EXTRA_STATE . . . . .	4727
7.654.1.11	FINISH_STATE . . . . .	4727
7.654.1.12	Freq . . . . .	4727
7.654.1.13	GZIP . . . . .	4727
7.654.1.14	HCRC_STATE . . . . .	4727
7.654.1.15	HEAP_SIZE . . . . .	4727
7.654.1.16	NIT_STATE . . . . .	4727
7.654.1.17	L_CODES . . . . .	4727



7.654.1.1	Len . . . . .	4727
7.654.1.1	LENGTH_CODES . . . . .	4727
7.654.1.2	LITERALS . . . . .	4727
7.654.1.2	MAX_BITS . . . . .	4727
7.654.1.2	MAX_DIST . . . . .	4727
7.654.1.2	max_insert_length . . . . .	4727
7.654.1.2	MIN_LOOKAHEAD . . . . .	4727
7.654.1.2	NAME_STATE . . . . .	4727
7.654.1.2	put_byte . . . . .	4727
7.654.1.2	WIN_INIT . . . . .	4727
7.654.2	Typedef Documentation . . . . .	4727
7.654.2.1	ct_data . . . . .	4727
7.654.2.2	deflate_state . . . . .	4727
7.654.2.3	IPos . . . . .	4727
7.654.2.4	Pos . . . . .	4727
7.654.2.5	Posf . . . . .	4727
7.654.2.6	static_tree_desc . . . . .	4727
7.654.2.7	tree_desc . . . . .	4727
7.654.3	Function Documentation . . . . .	4727
7.654.3.1	OF . . . . .	4727
7.654.3.2	OF . . . . .	4727
7.654.3.3	OF . . . . .	4727
7.654.4	Variable Documentation . . . . .	4727
7.654.4.1	_dist_code . . . . .	4727
7.654.4.2	_length_code . . . . .	4727
7.655	src/main/decaf/internal/util/zip/gzguts.h File Reference . . . . .	4728
7.655.1	Define Documentation . . . . .	4729
7.655.1.1	COPY . . . . .	4729
7.655.1.2	GT_OFF . . . . .	4729
7.655.1.3	GZ_APPEND . . . . .	4729
7.655.1.4	GZ_NONE . . . . .	4729
7.655.1.5	GZ_READ . . . . .	4729
7.655.1.6	GZ_WRITE . . . . .	4729
7.655.1.7	GZBUFSIZE . . . . .	4729
7.655.1.8	GZIP . . . . .	4729
7.655.1.9	local . . . . .	4729

7.655.1.1	LOOK	4729
7.655.1.1	ZLIB_INTERNAL	4729
7.655.1.2	zsterror	4729
7.655.2	Typedef Documentation	4729
7.655.2.1	gz_statep	4729
7.655.3	Function Documentation	4729
7.655.3.1	OF	4729
7.655.3.2	OF	4729
7.655.3.3	OF	4729
7.655.3.4	OF	4729
7.655.3.5	OF	4729
7.655.3.6	OF	4729
7.655.3.7	OF	4729
7.656src/main/decaf/internal/util/zip/inffast.h	File Reference	4730
7.656.1	Function Documentation	4730
7.656.1.1	OF	4730
7.657src/main/decaf/internal/util/zip/inffixed.h	File Reference	4731
7.658src/main/decaf/internal/util/zip/inflate.h	File Reference	4732
7.658.1	Define Documentation	4732
7.658.1.1	GUNZIP	4732
7.658.2	Enumeration Type Documentation	4732
7.658.2.1	inflate_mode	4732
7.659src/main/decaf/internal/util/zip/inftrees.h	File Reference	4734
7.659.1	Define Documentation	4734
7.659.1.1	ENOUGH	4734
7.659.1.2	ENOUGH_DISTS	4734
7.659.1.3	ENOUGH_LENS	4734
7.659.2	Enumeration Type Documentation	4734
7.659.2.1	codetype	4734
7.659.3	Function Documentation	4734
7.659.3.1	OF	4734
7.660src/main/decaf/internal/util/zip/trees.h	File Reference	4735
7.660.1	Variable Documentation	4735
7.660.1.1	_dist_code	4735
7.660.1.2	_length_code	4735
7.660.1.3	base_dist	4736

7.660.1.4 base_length . . . . .	4736
7.660.1.5 static_dtree . . . . .	4736
7.660.1.6 static_ltree . . . . .	4736
7.661src/main/decaf/internal/util/zip/zconf.h File Reference . . . . .	4737
7.661.1 Define Documentation . . . . .	4738
7.661.1.1 const . . . . .	4738
7.661.1.2 MAX_MEM_LEVEL . . . . .	4738
7.661.1.3 MAX_WBITS . . . . .	4738
7.661.1.4 OF . . . . .	4738
7.661.1.5 SEEK_CUR . . . . .	4738
7.661.1.6 SEEK_END . . . . .	4738
7.661.1.7 SEEK_SET . . . . .	4738
7.661.1.8 z_off64_t . . . . .	4738
7.661.1.9 z_off_t . . . . .	4738
7.661.1.10ZEXTERN . . . . .	4738
7.661.2 Typedef Documentation . . . . .	4738
7.661.2.1 Byte . . . . .	4738
7.661.2.2 Bytef . . . . .	4738
7.661.2.3 charf . . . . .	4738
7.661.2.4 intf . . . . .	4738
7.661.2.5 uInt . . . . .	4738
7.661.2.6 uIntf . . . . .	4738
7.661.2.7 uLong . . . . .	4738
7.661.2.8 uLongf . . . . .	4738
7.661.2.9 voidp . . . . .	4738
7.661.2.10voidpc . . . . .	4738
7.661.2.11voidpf . . . . .	4738
7.662src/main/decaf/internal/util/zip/zlib.h File Reference . . . . .	4739
7.662.1 Define Documentation . . . . .	4741
7.662.1.1 deflateInit . . . . .	4741
7.662.1.2 deflateInit2 . . . . .	4741
7.662.1.3 inflateBackInit . . . . .	4741
7.662.1.4 inflateInit . . . . .	4742
7.662.1.5 inflateInit2 . . . . .	4742
7.662.1.6 Z_ASCII . . . . .	4742
7.662.1.7 Z_BEST_COMPRESSION . . . . .	4742

7.662.1.8 Z_BEST_SPEED . . . . .	4742
7.662.1.9 Z_BINARY . . . . .	4742
7.662.1.10 Z_BLOCK . . . . .	4742
7.662.1.11 Z_BUF_ERROR . . . . .	4742
7.662.1.12 Z_DATA_ERROR . . . . .	4742
7.662.1.13 Z_DEFAULT_COMPRESSION . . . . .	4742
7.662.1.14 Z_DEFAULT_STRATEGY . . . . .	4742
7.662.1.15 Z_DEFLATED . . . . .	4742
7.662.1.16 Z_ERRNO . . . . .	4742
7.662.1.17 Z_FILTERED . . . . .	4742
7.662.1.18 Z_FINISH . . . . .	4742
7.662.1.19 Z_FIXED . . . . .	4742
7.662.1.20 Z_FULL_FLUSH . . . . .	4742
7.662.1.21 Z_HUFFMAN_ONLY . . . . .	4742
7.662.1.22 Z_MEM_ERROR . . . . .	4742
7.662.1.23 Z_NEED_DICT . . . . .	4742
7.662.1.24 Z_NO_COMPRESSION . . . . .	4742
7.662.1.25 Z_NO_FLUSH . . . . .	4742
7.662.1.26 Z_NULL . . . . .	4742
7.662.1.27 Z_OK . . . . .	4742
7.662.1.28 Z_PARTIAL_FLUSH . . . . .	4742
7.662.1.29 Z_RLE . . . . .	4742
7.662.1.30 Z_STREAM_END . . . . .	4742
7.662.1.31 Z_STREAM_ERROR . . . . .	4742
7.662.1.32 Z_SYNC_FLUSH . . . . .	4742
7.662.1.33 Z_TEXT . . . . .	4742
7.662.1.34 Z_TREES . . . . .	4742
7.662.1.35 Z_UNKNOWN . . . . .	4742
7.662.1.36 Z_VERSION_ERROR . . . . .	4742
7.662.1.37 ZLIB_VER_MAJOR . . . . .	4742
7.662.1.38 ZLIB_VER_MINOR . . . . .	4742
7.662.1.39 ZLIB_VER_REVISION . . . . .	4742
7.662.1.40 ZLIB_VER_SUBREVISION . . . . .	4742
7.662.1.41 ZLIB_VERNUM . . . . .	4742
7.662.1.42 lib_version . . . . .	4742
7.662.1.43 ZLIB_VERSION . . . . .	4742

7.662.2 Typedef Documentation . . . . .	4742
7.662.2.1 gz_header . . . . .	4742
7.662.2.2 gz_headerp . . . . .	4742
7.662.2.3 gzFile . . . . .	4742
7.662.2.4 OF . . . . .	4742
7.662.2.5 z_stream . . . . .	4742
7.662.2.6 z_streamp . . . . .	4742
7.662.3 Function Documentation . . . . .	4742
7.662.3.1 OF . . . . .	4742
7.662.3.2 OF . . . . .	4742
7.662.3.3 OF . . . . .	4742
7.662.3.4 OF . . . . .	4742
7.662.3.5 OF . . . . .	4742
7.662.3.6 OF . . . . .	4742
7.662.3.7 OF . . . . .	4742
7.662.3.8 OF . . . . .	4742
7.662.3.9 OF . . . . .	4742
7.662.3.10OF . . . . .	4742
7.662.3.11OF . . . . .	4742
7.662.3.12OF . . . . .	4742
7.662.3.13OF . . . . .	4742
7.662.3.14OF . . . . .	4742
7.662.3.15OF . . . . .	4742
7.662.3.16OF . . . . .	4742
7.662.3.17OF . . . . .	4742
7.662.3.18OF . . . . .	4742
7.662.3.19OF . . . . .	4742
7.662.3.20OF . . . . .	4742
7.662.3.21OF . . . . .	4742
7.662.3.22OF . . . . .	4742
7.662.3.23OF . . . . .	4742
7.662.3.24OF . . . . .	4742
7.662.3.25OF . . . . .	4742
7.662.3.26OF . . . . .	4742
7.662.3.27OF . . . . .	4742
7.662.3.28OF . . . . .	4742

7.662.3.29OF . . . . .	4742
7.662.3.30OF . . . . .	4742
7.662.3.31OF . . . . .	4742
7.662.3.32OF . . . . .	4742
7.662.3.33OF . . . . .	4742
7.662.3.34OF . . . . .	4742
7.662.3.35OF . . . . .	4742
7.662.3.36OF . . . . .	4742
7.662.3.37OF . . . . .	4742
7.662.3.38OF . . . . .	4742
7.662.3.39OF . . . . .	4742
7.662.3.40OF . . . . .	4742
7.662.3.41OF . . . . .	4742
7.662.3.42OF . . . . .	4742
7.663src/main/decaf/internal/util/zip/zutil.h File Reference . . . . .	4743
7.663.1 Define Documentation . . . . .	4745
7.663.1.1 Assert . . . . .	4745
7.663.1.2 DEF_MEM_LEVEL . . . . .	4745
7.663.1.3 DEF_WBITS . . . . .	4745
7.663.1.4 DYN_TREES . . . . .	4745
7.663.1.5 ERR_MSG . . . . .	4745
7.663.1.6 ERR_RETURN . . . . .	4745
7.663.1.7 F_OPEN . . . . .	4745
7.663.1.8 MAX_MATCH . . . . .	4745
7.663.1.9 MIN_MATCH . . . . .	4745
7.663.1.10OS_CODE . . . . .	4745
7.663.1.11PRESET_DICT . . . . .	4745
7.663.1.12STATIC_TREES . . . . .	4745
7.663.1.13STORED_BLOCK . . . . .	4745
7.663.1.14Trace . . . . .	4745
7.663.1.15Tracec . . . . .	4745
7.663.1.16Tracecv . . . . .	4745
7.663.1.17Tracev . . . . .	4745
7.663.1.18Tracevv . . . . .	4745
7.663.1.19TRY_FREE . . . . .	4745
7.663.1.20ZALLOC . . . . .	4745

7.663.1.2IZFREE . . . . .	4745
7.663.1.2ZLIB_INTERNAL . . . . .	4745
7.663.2 Typedef Documentation . . . . .	4745
7.663.2.1 uch . . . . .	4745
7.663.2.2 uchf . . . . .	4745
7.663.2.3 ulg . . . . .	4745
7.663.2.4 ush . . . . .	4745
7.663.2.5 ushf . . . . .	4745
7.663.3 Function Documentation . . . . .	4745
7.663.3.1 OF . . . . .	4745
7.663.3.2 OF . . . . .	4745
7.663.3.3 OF . . . . .	4745
7.663.3.4 OF . . . . .	4745
7.663.3.5 OF . . . . .	4745
7.663.3.6 OF . . . . .	4745
7.663.4 Variable Documentation . . . . .	4745
7.663.4.1 z_errmsg . . . . .	4745
7.664src/main/decaf/io/BlockingByteArrayInputStream.h File Reference . . . . .	4746
7.665src/main/decaf/io/BufferedInputStream.h File Reference . . . . .	4747
7.666src/main/decaf/io/BufferedOutputStream.h File Reference . . . . .	4748
7.667src/main/decaf/io/ByteArrayInputStream.h File Reference . . . . .	4749
7.668src/main/decaf/io/ByteArrayOutputStream.h File Reference . . . . .	4750
7.669src/main/decaf/io/DataInput.h File Reference . . . . .	4751
7.670src/main/decaf/io/DataInputStream.h File Reference . . . . .	4752
7.671src/main/decaf/io/DataOutput.h File Reference . . . . .	4753
7.672src/main/decaf/io/DataOutputStream.h File Reference . . . . .	4754
7.673src/main/decaf/io/EOFException.h File Reference . . . . .	4755
7.674src/main/decaf/io/FileDescriptor.h File Reference . . . . .	4756
7.675src/main/decaf/io/FilterInputStream.h File Reference . . . . .	4757
7.676src/main/decaf/io/FilterOutputStream.h File Reference . . . . .	4758
7.677src/main/decaf/io/Flushable.h File Reference . . . . .	4759
7.678src/main/decaf/io/InputStream.h File Reference . . . . .	4760
7.679src/main/decaf/io/InputStreamReader.h File Reference . . . . .	4761
7.680src/main/decaf/io/InterruptedIOException.h File Reference . . . . .	4762
7.681src/main/decaf/io/IOException.h File Reference . . . . .	4763
7.682src/main/decaf/io/OutputStream.h File Reference . . . . .	4764

7.683src/main/decaf/io/OutputStreamWriter.h File Reference . . . . .	4765
7.684src/main/decaf/io/PushbackInputStream.h File Reference . . . . .	4766
7.685src/main/decaf/io/Reader.h File Reference . . . . .	4767
7.686src/main/decaf/io/UnsupportedEncodingException.h File Reference . . . . .	4768
7.687src/main/decaf/io/UTFDataFormatException.h File Reference . . . . .	4769
7.688src/main/decaf/io/Writer.h File Reference . . . . .	4770
7.689src/main/decaf/lang/Appendable.h File Reference . . . . .	4771
7.690src/main/decaf/lang/ArrayPointer.h File Reference . . . . .	4772
7.691src/main/decaf/lang/Boolean.h File Reference . . . . .	4774
7.692src/main/decaf/lang/Byte.h File Reference . . . . .	4775
7.693src/main/decaf/lang/Character.h File Reference . . . . .	4776
7.694src/main/decaf/lang/CharSequence.h File Reference . . . . .	4777
7.695src/main/decaf/lang/Comparable.h File Reference . . . . .	4778
7.696src/main/decaf/lang/Double.h File Reference . . . . .	4779
7.697src/main/decaf/lang/Exception.h File Reference . . . . .	4780
7.698src/main/decaf/lang/exceptions/ClassCastException.h File Reference . . . . .	4781
7.699src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference . . . . .	4782
7.700src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference . . . . .	4783
7.701src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference . . . . .	4784
7.702src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference . . . . .	4785
7.703src/main/decaf/lang/exceptions/InterruptedException.h File Reference . . . . .	4786
7.704src/main/decaf/lang/exceptions/InvalidStateException.h File Reference . . . . .	4787
7.705src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference . . . . .	4788
7.706src/main/decaf/lang/exceptions/NullPointerException.h File Reference . . . . .	4789
7.707src/main/decaf/lang/exceptions/NumberFormatException.h File Reference . . . . .	4790
7.708src/main/decaf/lang/exceptions/RuntimeException.h File Reference . . . . .	4791
7.709src/main/decaf/lang/Float.h File Reference . . . . .	4792
7.710src/main/decaf/lang/Integer.h File Reference . . . . .	4793
7.711src/main/decaf/lang/Iterable.h File Reference . . . . .	4794
7.712src/main/decaf/lang/Long.h File Reference . . . . .	4795
7.713src/main/decaf/lang/Math.h File Reference . . . . .	4796
7.714src/main/decaf/lang/Number.h File Reference . . . . .	4797
7.715src/main/decaf/lang/Pointer.h File Reference . . . . .	4798
7.716src/main/decaf/lang/Readable.h File Reference . . . . .	4799
7.717src/main/decaf/lang/Runnable.h File Reference . . . . .	4800
7.718src/main/decaf/lang/Runtime.h File Reference . . . . .	4801



7.719src/main/decaf/lang/Short.h File Reference . . . . .	4802
7.720src/main/decaf/lang/String.h File Reference . . . . .	4803
7.721src/main/decaf/lang/System.h File Reference . . . . .	4804
7.722src/main/decaf/lang/Thread.h File Reference . . . . .	4805
7.723src/main/decaf/lang/ThreadGroup.h File Reference . . . . .	4806
7.724src/main/decaf/lang/Throwable.h File Reference . . . . .	4807
7.725src/main/decaf/net/BindException.h File Reference . . . . .	4808
7.726src/main/decaf/net/ConnectException.h File Reference . . . . .	4809
7.727src/main/decaf/net/HttpRetryException.h File Reference . . . . .	4810
7.728src/main/decaf/net/Inet4Address.h File Reference . . . . .	4811
7.729src/main/decaf/net/Inet6Address.h File Reference . . . . .	4812
7.730src/main/decaf/net/InetAddress.h File Reference . . . . .	4813
7.731src/main/decaf/net/InetSocketAddress.h File Reference . . . . .	4814
7.732src/main/decaf/net/MalformedURLException.h File Reference . . . . .	4815
7.733src/main/decaf/net/NoRouteToHostException.h File Reference . . . . .	4816
7.734src/main/decaf/net/PortUnreachableException.h File Reference . . . . .	4817
7.735src/main/decaf/net/ProtocolException.h File Reference . . . . .	4818
7.736src/main/decaf/net/ServerSocket.h File Reference . . . . .	4819
7.737src/main/decaf/net/ServerSocketFactory.h File Reference . . . . .	4820
7.738src/main/decaf/net/Socket.h File Reference . . . . .	4821
7.739src/main/decaf/net/SocketAddress.h File Reference . . . . .	4822
7.740src/main/decaf/net/SocketError.h File Reference . . . . .	4823
7.741src/main/decaf/net/SocketException.h File Reference . . . . .	4824
7.742src/main/decaf/net/SocketFactory.h File Reference . . . . .	4825
7.743src/main/decaf/net/SocketImpl.h File Reference . . . . .	4826
7.744src/main/decaf/net/SocketImplFactory.h File Reference . . . . .	4827
7.745src/main/decaf/net/SocketOptions.h File Reference . . . . .	4828
7.746src/main/decaf/net/SocketTimeoutException.h File Reference . . . . .	4829
7.747src/main/decaf/net/ssl/SSLContext.h File Reference . . . . .	4830
7.748src/main/decaf/net/ssl/SSLContextSpi.h File Reference . . . . .	4831
7.749src/main/decaf/net/ssl/SSLParameters.h File Reference . . . . .	4832
7.750src/main/decaf/net/ssl/SSLServerSocket.h File Reference . . . . .	4833
7.751src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference . . . . .	4834
7.752src/main/decaf/net/ssl/SSLSocket.h File Reference . . . . .	4835
7.753src/main/decaf/net/ssl/SSLSocketFactory.h File Reference . . . . .	4836
7.754src/main/decaf/net/UnknownHostException.h File Reference . . . . .	4837

7.755src/main/decaf/net/UnknownServiceException.h File Reference . . . . .	4838
7.756src/main/decaf/net/URL.h File Reference . . . . .	4839
7.757src/main/decaf/net/URISyntaxException.h File Reference . . . . .	4840
7.758src/main/decaf/net/URL.h File Reference . . . . .	4841
7.759src/main/decaf/net/URLDecoder.h File Reference . . . . .	4842
7.760src/main/decaf/net/URLEncoder.h File Reference . . . . .	4843
7.761src/main/decaf/nio/Buffer.h File Reference . . . . .	4844
7.762src/main/decaf/nio/BufferOverflowException.h File Reference . . . . .	4845
7.763src/main/decaf/nio/BufferUnderflowException.h File Reference . . . . .	4846
7.764src/main/decaf/nio/ByteBuffer.h File Reference . . . . .	4847
7.765src/main/decaf/nio/CharBuffer.h File Reference . . . . .	4848
7.766src/main/decaf/nio/DoubleBuffer.h File Reference . . . . .	4849
7.767src/main/decaf/nio/FloatBuffer.h File Reference . . . . .	4850
7.768src/main/decaf/nio/IntBuffer.h File Reference . . . . .	4851
7.769src/main/decaf/nio/InvalidMarkException.h File Reference . . . . .	4852
7.770src/main/decaf/nio/LongBuffer.h File Reference . . . . .	4853
7.771src/main/decaf/nio/ReadOnlyBufferException.h File Reference . . . . .	4854
7.772src/main/decaf/nio/ShortBuffer.h File Reference . . . . .	4855
7.773src/main/decaf/security/auth/x500/X500Principal.h File Reference . . . . .	4856
7.774src/main/decaf/security/cert/Certificate.h File Reference . . . . .	4857
7.775src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . . . .	4858
7.776src/main/decaf/security/cert/CertificateException.h File Reference . . . . .	4859
7.777src/main/decaf/security/cert/CertificateExpiredException.h File Reference . . . . .	4860
7.778src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . . . . .	4861
7.779src/main/decaf/security/cert/CertificateParsingException.h File Reference . . . . .	4862
7.780src/main/decaf/security/cert/X509Certificate.h File Reference . . . . .	4863
7.781src/main/decaf/security/GeneralSecurityException.h File Reference . . . . .	4864
7.782src/main/decaf/security/InvalidKeyException.h File Reference . . . . .	4865
7.783src/main/decaf/security/Key.h File Reference . . . . .	4866
7.784src/main/decaf/security/KeyException.h File Reference . . . . .	4867
7.785src/main/decaf/security/KeyManagementException.h File Reference . . . . .	4868
7.786src/main/decaf/security/NoSuchAlgorithmException.h File Reference . . . . .	4869
7.787src/main/decaf/security/NoSuchProviderException.h File Reference . . . . .	4870
7.788src/main/decaf/security/Principal.h File Reference . . . . .	4871
7.789src/main/decaf/security/PublicKey.h File Reference . . . . .	4872
7.790src/main/decaf/security/SecureRandom.h File Reference . . . . .	4873

7.791src/main/decaf/security/SecureRandomSpi.h File Reference . . . . .	4874
7.792src/main/decaf/security/SignatureException.h File Reference . . . . .	4875
7.793src/main/decaf/util/AbstractCollection.h File Reference . . . . .	4876
7.794src/main/decaf/util/AbstractList.h File Reference . . . . .	4877
7.795src/main/decaf/util/AbstractMap.h File Reference . . . . .	4878
7.796src/main/decaf/util/AbstractQueue.h File Reference . . . . .	4879
7.797src/main/decaf/util/AbstractSequentialList.h File Reference . . . . .	4880
7.798src/main/decaf/util/AbstractSet.h File Reference . . . . .	4881
7.799src/main/decaf/util/Collection.h File Reference . . . . .	4882
7.800src/main/decaf/util/Comparator.h File Reference . . . . .	4883
7.801src/main/decaf/util/comparators/Less.h File Reference . . . . .	4884
7.802src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference . . . . .	4885
7.803src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference . . . . .	4886
7.804src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference . . . . .	4887
7.805src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference . . . . .	4888
7.806src/main/decaf/util/concurrent/BlockingQueue.h File Reference . . . . .	4889
7.807src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference . . . . .	4890
7.808src/main/decaf/util/concurrent/Callable.h File Reference . . . . .	4891
7.809src/main/decaf/util/concurrent/CancellationException.h File Reference . . . . .	4892
7.810src/main/decaf/util/concurrent/Concurrent.h File Reference . . . . .	4893
7.810.1 Define Documentation . . . . .	4893
7.810.1.1 synchronized . . . . .	4893
7.810.1.2 WAIT_INFINITE . . . . .	4893
7.811src/main/decaf/util/concurrent/ConcurrentMap.h File Reference . . . . .	4894
7.812src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference . . . . .	4895
7.813src/main/decaf/util/concurrent/CountDownLatch.h File Reference . . . . .	4896
7.814src/main/decaf/util/concurrent/Delayed.h File Reference . . . . .	4897
7.815src/main/decaf/util/concurrent/ExecutionException.h File Reference . . . . .	4898
7.816src/main/decaf/util/concurrent/Executor.h File Reference . . . . .	4899
7.817src/main/decaf/util/concurrent/ExecutorService.h File Reference . . . . .	4900
7.818src/main/decaf/util/concurrent/Future.h File Reference . . . . .	4901
7.819src/main/decaf/util/concurrent/Lock.h File Reference . . . . .	4902
7.820src/main/decaf/util/concurrent/locks/Lock.h File Reference . . . . .	4903
7.821src/main/decaf/util/concurrent/locks/Condition.h File Reference . . . . .	4904
7.822src/main/decaf/util/concurrent/locks/LockSupport.h File Reference . . . . .	4905
7.823src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference . . . . .	4906

7.824src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference . . . . .	4907
7.825src/main/decaf/util/concurrent/Mutex.h File Reference . . . . .	4908
7.826src/main/decaf/util/concurrent/PooledThread.h File Reference . . . . .	4909
7.827src/main/decaf/util/concurrent/PooledThreadListener.h File Reference . . . . .	4910
7.828src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference . . .	4911
7.829src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference . . .	4912
7.830src/main/decaf/util/concurrent/Semaphore.h File Reference . . . . .	4913
7.831src/main/decaf/util/concurrent/Synchronizable.h File Reference . . . . .	4914
7.832src/main/decaf/util/concurrent/SynchronousQueue.h File Reference . . . . .	4915
7.833src/main/decaf/util/concurrent/TaskListener.h File Reference . . . . .	4916
7.834src/main/decaf/util/concurrent/ThreadFactory.h File Reference . . . . .	4917
7.835src/main/decaf/util/concurrent/ThreadPool.h File Reference . . . . .	4918
7.836src/main/decaf/util/concurrent/TimeoutException.h File Reference . . . . .	4919
7.837src/main/decaf/util/concurrent/TimeUnit.h File Reference . . . . .	4920
7.838src/main/decaf/util/Date.h File Reference . . . . .	4921
7.839src/main/decaf/util/Iterator.h File Reference . . . . .	4922
7.840src/main/decaf/util/List.h File Reference . . . . .	4923
7.841src/main/decaf/util/ListIterator.h File Reference . . . . .	4924
7.842src/main/decaf/util/logging/ConsoleHandler.h File Reference . . . . .	4925
7.843src/main/decaf/util/logging/ErrorHandler.h File Reference . . . . .	4926
7.844src/main/decaf/util/logging/Filter.h File Reference . . . . .	4927
7.845src/main/decaf/util/logging/Formatter.h File Reference . . . . .	4928
7.846src/main/decaf/util/logging/Handler.h File Reference . . . . .	4929
7.847src/main/decaf/util/logging/Level.h File Reference . . . . .	4930
7.848src/main/decaf/util/logging/Logger.h File Reference . . . . .	4931
7.849src/main/decaf/util/logging/LoggerCommon.h File Reference . . . . .	4932
7.850src/main/decaf/util/logging/LoggerDefines.h File Reference . . . . .	4933
7.850.1 Define Documentation . . . . .	4933
7.850.1.1 LOGDECAF_DEBUG . . . . .	4933
7.850.1.2 LOGDECAF_DEBUG_1 . . . . .	4933
7.850.1.3 LOGDECAF_DECLARE . . . . .	4934
7.850.1.4 LOGDECAF_DECLARE_LOCAL . . . . .	4934
7.850.1.5 LOGDECAF_ERROR . . . . .	4934
7.850.1.6 LOGDECAF_FATAL . . . . .	4934
7.850.1.7 LOGDECAF_INFO . . . . .	4934
7.850.1.8 LOGDECAF_INITIALIZE . . . . .	4934

7.850.1.9 LOGDECAF_ WARN . . . . .	4934
7.851src/main/decaf/util/logging/LoggerHierarchy.h File Reference . . . . .	4935
7.852src/main/decaf/util/logging/LogManager.h File Reference . . . . .	4936
7.853src/main/decaf/util/logging/LogRecord.h File Reference . . . . .	4937
7.854src/main/decaf/util/logging/LogWriter.h File Reference . . . . .	4938
7.855src/main/decaf/util/logging/MarkBlockLogger.h File Reference . . . . .	4939
7.856src/main/decaf/util/logging/PropertiesChangeListener.h File Reference . . . . .	4940
7.857src/main/decaf/util/logging/SimpleFormatter.h File Reference . . . . .	4941
7.858src/main/decaf/util/logging/SimpleLogger.h File Reference . . . . .	4942
7.859src/main/decaf/util/logging/StreamHandler.h File Reference . . . . .	4943
7.860src/main/decaf/util/logging/XMLFormatter.h File Reference . . . . .	4944
7.861src/main/decaf/util/Map.h File Reference . . . . .	4945
7.862src/main/decaf/util/PriorityQueue.h File Reference . . . . .	4946
7.863src/main/decaf/util/Properties.h File Reference . . . . .	4947
7.864src/main/decaf/util/Random.h File Reference . . . . .	4948
7.865src/main/decaf/util/Set.h File Reference . . . . .	4949
7.866src/main/decaf/util/StlList.h File Reference . . . . .	4950
7.867src/main/decaf/util/StlMap.h File Reference . . . . .	4951
7.868src/main/decaf/util/StlQueue.h File Reference . . . . .	4952
7.869src/main/decaf/util/StlSet.h File Reference . . . . .	4953
7.870src/main/decaf/util/StringTokenizer.h File Reference . . . . .	4954
7.871src/main/decaf/util/Timer.h File Reference . . . . .	4955
7.872src/main/decaf/util/TimerTask.h File Reference . . . . .	4956
7.873src/main/decaf/util/UUID.h File Reference . . . . .	4957
7.874src/main/decaf/util/zip/Adler32.h File Reference . . . . .	4958
7.875src/main/decaf/util/zip/CheckedInputStream.h File Reference . . . . .	4959
7.876src/main/decaf/util/zip/CheckedOutputStream.h File Reference . . . . .	4960
7.877src/main/decaf/util/zip/Checksum.h File Reference . . . . .	4961
7.878src/main/decaf/util/zip/CRC32.h File Reference . . . . .	4962
7.879src/main/decaf/util/zip/DataFormatException.h File Reference . . . . .	4963
7.880src/main/decaf/util/zip/Deflater.h File Reference . . . . .	4964
7.881src/main/decaf/util/zip/DeflaterOutputStream.h File Reference . . . . .	4965
7.882src/main/decaf/util/zip/Inflater.h File Reference . . . . .	4966
7.883src/main/decaf/util/zip/InflaterInputStream.h File Reference . . . . .	4967
7.884src/main/decaf/util/zip/ZipException.h File Reference . . . . .	4968



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>activemq</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	85
<b>activemq::cmsutil</b> . . . . .	86
<b>activemq::commands</b> . . . . .	87
<b>activemq::core</b> . . . . .	89
<b>activemq::core::policies</b> . . . . .	91
<b>activemq::exceptions</b> . . . . .	92
<b>activemq::io</b> . . . . .	93
<b>activemq::library</b> . . . . .	94
<b>activemq::state</b> . . . . .	95
<b>activemq::threads</b> . . . . .	96
<b>activemq::transport</b> . . . . .	97
<b>activemq::transport::correlator</b> . . . . .	98
<b>activemq::transport::failover</b> . . . . .	99
<b>activemq::transport::inactivity</b> . . . . .	100
<b>activemq::transport::logging</b> . . . . .	101
<b>activemq::transport::mock</b> . . . . .	102
<b>activemq::transport::tcp</b> . . . . .	103
<b>activemq::util</b> . . . . .	104
<b>activemq::wireformat</b> . . . . .	105
<b>activemq::wireformat::openwire</b> . . . . .	106
<b>activemq::wireformat::openwire::marshal</b> . . . . .	107
<b>activemq::wireformat::openwire::marshal::v1</b> . . . . .	108
<b>activemq::wireformat::openwire::marshal::v2</b> . . . . .	113
<b>activemq::wireformat::openwire::marshal::v3</b> . . . . .	118
<b>activemq::wireformat::openwire::marshal::v4</b> . . . . .	123
<b>activemq::wireformat::openwire::marshal::v5</b> . . . . .	128
<b>activemq::wireformat::openwire::marshal::v6</b> . . . . .	133
<b>activemq::wireformat::openwire::utils</b> . . . . .	138
<b>activemq::wireformat::stomp</b> . . . . .	139
<b>cms</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	140

<b>decaf</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	143
<b>decaf::internal</b> . . . . .	144
<b>decaf::internal::io</b> . . . . .	145
<b>decaf::internal::net</b> . . . . .	146
<b>decaf::internal::net::ssl</b> . . . . .	147
<b>decaf::internal::net::ssl::openssl</b> . . . . .	148
<b>decaf::internal::net::tcp</b> . . . . .	149
<b>decaf::internal::nio</b> . . . . .	150
<b>decaf::internal::security</b> . . . . .	151
<b>decaf::internal::util</b> . . . . .	152
<b>decaf::internal::util::concurrent</b> . . . . .	153
<b>decaf::io</b> . . . . .	154
<b>decaf::lang</b> . . . . .	156
<b>decaf::lang::exceptions</b> . . . . .	159
<b>decaf::net</b> . . . . .	160
<b>decaf::net::ssl</b> . . . . .	162
<b>decaf::nio</b> . . . . .	163
<b>decaf::security</b> . . . . .	164
<b>decaf::security::auth</b> . . . . .	165
<b>decaf::security::auth::x500</b> . . . . .	166
<b>decaf::security::cert</b> . . . . .	167
<b>decaf::util</b> . . . . .	168
<b>decaf::util::comparators</b> . . . . .	170
<b>decaf::util::concurrent</b> . . . . .	171
<b>decaf::util::concurrent::atomic</b> . . . . .	173
<b>decaf::util::concurrent::locks</b> . . . . .	174
<b>decaf::util::logging</b> . . . . .	175
<b>decaf::util::zip</b> . . . . .	177
<b>std</b> . . . . .	178



## Chapter 2

# Data Structure Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler . . . . .	203
activemq::core::ActiveMQConstants . . . . .	307
activemq::library::ActiveMQCPP . . . . .	320
activemq::core::ActiveMQTransactionContext . . . . .	719
decaf::lang::Appendable . . . . .	725
decaf::io::Writer . . . . .	4021
decaf::io::OutputStreamWriter . . . . .	2915
decaf::nio::CharBuffer . . . . .	1119
decaf::internal::nio::CharArrayBuffer . . . . .	1108
decaf::internal::AprPool . . . . .	728
decaf::lang::ArrayPointer< T, REFCOUNTER > . . . . .	730
decaf::util::concurrent::atomic::AtomicBoolean . . . . .	738
decaf::util::concurrent::atomic::AtomicRefCounter . . . . .	746
decaf::util::concurrent::atomic::AtomicReference< T > . . . . .	749
binary_function . . . . .	835
decaf::util::Comparator< ArrayPointer< T, R > > . . . . .	1217
decaf::lang::ArrayPointerComparator< T, R > . . . . .	737
decaf::util::Comparator< Pointer< T, R > > . . . . .	1217
decaf::lang::PointerComparator< T, R > . . . . .	2954
std::less< decaf::lang::ArrayPointer< T > > . . . . .	2330
std::less< decaf::lang::Pointer< T > > . . . . .	2331
activemq::wireformat::openwire::utils::BooleanStream . . . . .	857
decaf::nio::Buffer . . . . .	928
decaf::nio::ByteBuffer . . . . .	1031
decaf::internal::nio::ByteBuffer . . . . .	991
decaf::nio::CharBuffer . . . . .	1119
decaf::nio::DoubleBuffer . . . . .	1809
decaf::internal::nio::DoubleArrayBuffer . . . . .	1799
decaf::nio::FloatBuffer . . . . .	1925
decaf::internal::nio::FloatArrayBuffer . . . . .	1915
decaf::nio::IntBuffer . . . . .	2066
decaf::internal::nio::IntArrayBuffer . . . . .	2056

decaf::nio::LongBuffer . . . . .	2444
decaf::internal::nio::LongArrayBuffer . . . . .	2434
decaf::nio::ShortBuffer . . . . .	3459
decaf::internal::nio::ShortArrayBuffer . . . . .	3449
decaf::internal::nio::BufferFactory . . . . .	942
decaf::internal::util::ByteArrayAdapter . . . . .	969
decaf::util::concurrent::Callable< V > . . . . .	1082
decaf::security::cert::Certificate . . . . .	1086
decaf::security::cert::X509Certificate . . . . .	4028
decaf::lang::CharSequence . . . . .	1135
decaf::lang::String . . . . .	3665
decaf::nio::CharBuffer . . . . .	1119
decaf::util::zip::Checksum . . . . .	1142
decaf::util::zip::Adler32 . . . . .	722
decaf::util::zip::CRC32 . . . . .	1524
cms::Closeable . . . . .	1148
activemq::commands::ActiveMQTempDestination . . . . .	576
activemq::commands::ActiveMQTempQueue . . . . .	604
activemq::commands::ActiveMQTempTopic . . . . .	633
cms::Connection . . . . .	1262
activemq::core::ActiveMQConnection . . . . .	273
cms::MessageConsumer . . . . .	2589
activemq::cmsutil::CachedConsumer . . . . .	1071
activemq::core::ActiveMQConsumer . . . . .	310
cms::MessageProducer . . . . .	2725
activemq::cmsutil::CachedProducer . . . . .	1075
activemq::core::ActiveMQProducer . . . . .	470
cms::QueueBrowser . . . . .	3153
activemq::core::ActiveMQQueueBrowser . . . . .	487
cms::Session . . . . .	3365
activemq::cmsutil::PooledSession . . . . .	2955
activemq::core::ActiveMQSession . . . . .	515
decaf::io::Closeable . . . . .	1149
activemq::transport::Transport . . . . .	3883
activemq::transport::CompositeTransport . . . . .	1226
activemq::transport::failover::FailoverTransport . . . . .	1873
activemq::transport::IOTransport . . . . .	2145
activemq::transport::mock::MockTransport . . . . .	2768
activemq::transport::TransportFilter . . . . .	3891
activemq::transport::correlator::ResponseCorrelator . . . . .	3291
activemq::transport::inactivity::InactivityMonitor . . . . .	2004
activemq::transport::logging::LoggingTransport . . . . .	2403
activemq::transport::tcp::TcpTransport . . . . .	3752
activemq::transport::tcp::SslTransport . . . . .	3574
activemq::wireformat::WireFormatNegotiator . . . . .	4016
activemq::wireformat::openwire::OpenWireFormatNegotiator . . . . .	2901
decaf::io::InputStream . . . . .	2043
decaf::internal::io::StandardInputStream . . . . .	3582
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream . . . . .	2882
decaf::internal::net::tcp::TcpSocketInputStream . . . . .	3747
decaf::io::BlockingByteArrayInputStream . . . . .	839

decaf::io::ByteArrayInputStream . . . . .	1020
decaf::io::FilterInputStream . . . . .	1894
activemq::io::LoggingInputStream . . . . .	2399
decaf::io::BufferedInputStream . . . . .	934
decaf::io::DataInputStream . . . . .	1566
decaf::io::PushbackInputStream . . . . .	3141
decaf::util::zip::CheckedInputStream . . . . .	1137
decaf::util::zip::InflaterInputStream . . . . .	2035
decaf::io::OutputStream . . . . .	2907
decaf::internal::io::StandardErrorOutputStream . . . . .	3579
decaf::internal::io::StandardOutputStream . . . . .	3584
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream . . . . .	2885
decaf::internal::net::tcp::TcpSocketOutputStream . . . . .	3750
decaf::io::ByteArrayOutputStream . . . . .	1028
decaf::io::FilterOutputStream . . . . .	1900
activemq::io::LoggingOutputStream . . . . .	2401
decaf::io::BufferedOutputStream . . . . .	940
decaf::io::DataOutputStream . . . . .	1579
decaf::util::zip::CheckedOutputStream . . . . .	1140
decaf::util::zip::DeflaterOutputStream . . . . .	1716
decaf::io::Reader . . . . .	3163
decaf::io::InputStreamReader . . . . .	2053
decaf::io::Writer . . . . .	4021
decaf::net::Socket . . . . .	3503
decaf::net::ssl::SSLSocket . . . . .	3563
decaf::internal::net::ssl::openssl::OpenSSLSocket . . . . .	2858
decaf::util::logging::Handler . . . . .	1978
decaf::util::logging::StreamHandler . . . . .	3649
decaf::util::logging::ConsoleHandler . . . . .	1399
activemq::cmsutil::CmsAccessor . . . . .	1153
activemq::cmsutil::CmsDestinationAccessor . . . . .	1157
activemq::cmsutil::CmsTemplate . . . . .	1170
cms::CMSException . . . . .	1160
cms::CMSSecurityException . . . . .	1169
cms::IllegalStateException . . . . .	1997
cms::InvalidClientIdException . . . . .	2130
cms::InvalidDestinationException . . . . .	2131
cms::InvalidSelectorException . . . . .	2138
cms::MessageEOFException . . . . .	2661
cms::MessageFormatException . . . . .	2662
cms::MessageNotReadableException . . . . .	2723
cms::MessageNotWriteableException . . . . .	2724
cms::UnsupportedOperationException . . . . .	3919
activemq::util::CMSExceptionSupport . . . . .	1163
cms::CMSProperties . . . . .	1165
activemq::util::ActiveMQProperties . . . . .	478
code . . . . .	1183
activemq::state::CommandVisitor . . . . .	1200
activemq::state::CommandVisitorAdapter . . . . .	1208
activemq::state::ConnectionStateTracker . . . . .	1392
decaf::lang::Comparable< T > . . . . .	1214

decaf::lang::Comparable< bool > . . . . .	1214
decaf::lang::Boolean . . . . .	850
decaf::lang::Comparable< Boolean > . . . . .	1214
decaf::lang::Boolean . . . . .	850
decaf::lang::Comparable< BrokerId > . . . . .	1214
activemq::commands::BrokerId . . . . .	869
decaf::lang::Comparable< Byte > . . . . .	1214
decaf::lang::Byte . . . . .	960
decaf::lang::Comparable< ByteBuffer > . . . . .	1214
decaf::nio::ByteBuffer . . . . .	1031
decaf::lang::Comparable< char > . . . . .	1214
decaf::lang::Character . . . . .	1100
decaf::lang::Comparable< Character > . . . . .	1214
decaf::lang::Character . . . . .	1100
decaf::lang::Comparable< CharBuffer > . . . . .	1214
decaf::nio::CharBuffer . . . . .	1119
decaf::lang::Comparable< ConnectionId > . . . . .	1214
activemq::commands::ConnectionId . . . . .	1327
decaf::lang::Comparable< ConsumerId > . . . . .	1214
activemq::commands::ConsumerId . . . . .	1430
decaf::lang::Comparable< Date > . . . . .	1214
decaf::util::Date . . . . .	1665
decaf::lang::Comparable< Delayed > . . . . .	1214
decaf::util::concurrent::Delayed . . . . .	1720
decaf::lang::Comparable< Double > . . . . .	1214
decaf::lang::Double . . . . .	1788
decaf::lang::Comparable< double > . . . . .	1214
decaf::lang::Double . . . . .	1788
decaf::lang::Comparable< DoubleBuffer > . . . . .	1214
decaf::nio::DoubleBuffer . . . . .	1809
decaf::lang::Comparable< float > . . . . .	1214
decaf::lang::Float . . . . .	1904
decaf::lang::Comparable< Float > . . . . .	1214
decaf::lang::Float . . . . .	1904
decaf::lang::Comparable< FloatBuffer > . . . . .	1214
decaf::nio::FloatBuffer . . . . .	1925
decaf::lang::Comparable< int > . . . . .	1214
decaf::lang::Integer . . . . .	2077
decaf::lang::Comparable< IntBuffer > . . . . .	1214
decaf::nio::IntBuffer . . . . .	2066
decaf::lang::Comparable< Integer > . . . . .	1214
decaf::lang::Integer . . . . .	2077
decaf::lang::Comparable< Level > . . . . .	1214
decaf::util::logging::Level . . . . .	2332
decaf::lang::Comparable< LocalTransactionId > . . . . .	1214
activemq::commands::LocalTransactionId . . . . .	2347

decaf::lang::Comparable< Long > . . . . .	1214
decaf::lang::Long . . . . .	2420
decaf::lang::Comparable< long long > . . . . .	1214
decaf::lang::Long . . . . .	2420
decaf::lang::Comparable< LongBuffer > . . . . .	1214
decaf::nio::LongBuffer . . . . .	2444
decaf::lang::Comparable< MessageId > . . . . .	1214
activemq::commands::MessageId . . . . .	2663
decaf::lang::Comparable< ProducerId > . . . . .	1214
activemq::commands::ProducerId . . . . .	3067
decaf::lang::Comparable< SessionId > . . . . .	1214
activemq::commands::SessionId . . . . .	3379
decaf::lang::Comparable< Short > . . . . .	1214
decaf::lang::Short . . . . .	3440
decaf::lang::Comparable< short > . . . . .	1214
decaf::lang::Short . . . . .	3440
decaf::lang::Comparable< ShortBuffer > . . . . .	1214
decaf::nio::ShortBuffer . . . . .	3459
decaf::lang::Comparable< TimeUnit > . . . . .	1214
decaf::util::concurrent::TimeUnit . . . . .	3807
decaf::lang::Comparable< TransactionId > . . . . .	1214
activemq::commands::TransactionId . . . . .	3819
activemq::commands::LocalTransactionId . . . . .	2347
activemq::commands::XATransactionId . . . . .	4031
decaf::lang::Comparable< unsigned char > . . . . .	1214
decaf::lang::Byte . . . . .	960
decaf::lang::Comparable< URI > . . . . .	1214
decaf::net::URI . . . . .	3921
decaf::lang::Comparable< UUID > . . . . .	1214
decaf::util::UUID . . . . .	3969
decaf::lang::Comparable< XATransactionId > . . . . .	1214
activemq::commands::XATransactionId . . . . .	4031
decaf::util::Comparator< T > . . . . .	1217
activemq::util::CompositeData . . . . .	1219
decaf::util::concurrent::locks::Condition . . . . .	1249
decaf::util::concurrent::ConditionHandle . . . . .	1255
decaf::internal::util::concurrent::ConditionImpl . . . . .	1257
cms::ConnectionFactory . . . . .	1324
activemq::core::ActiveMQConnectionFactory . . . . .	292
cms::ConnectionMetaData . . . . .	1385
activemq::core::ActiveMQConnectionMetaData . . . . .	303
activemq::state::ConnectionState . . . . .	1389
activemq::state::ConsumerState . . . . .	1492
decaf::util::concurrent::CountDownLatch . . . . .	1521
ct_data_s . . . . .	1527
decaf::io::DataInput . . . . .	1558
decaf::io::DataOutput . . . . .	1574
activemq::wireformat::openwire::marshal::DataStreamMarshaller . . . . .	1610

activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller . . . . .	807
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller . . .	337
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller . . . .	495
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	584
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	613
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	646
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller . . . .	703
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller . . . . .	779
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller . . . . .	912
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller . .	1280
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller . . .	1312
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller . . . .	1373
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller . . .	1418
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller . . . . .	1480
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller . . .	1509
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller . . . .	1744
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller . . . .	1956
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller . . . . .	2289
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller . . . . .	2581
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller . . .	2622
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2651
activemq::wireformat::openwire::marshal::v1::MessageMarshaller . . . . .	2713
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	213
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	253
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	378
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller .	405
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	450
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	556
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	675
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller . . . . .	2760
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller . . . . .	3060
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller . . . . .	3109
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller . . . . .	3210
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	3227
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller . . .	3258
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller . . . . .	3315
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	1543
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller . . .	1606
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1862
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller . .	2110
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller . . . . .	3419
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller . . . .	3481
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller . . . .	3855
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller . . . . .	880
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller . . . . .	1343
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller . . . . .	1447
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller . . . . .	1778
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller . . . .	2179
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller . . . .	2208
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller . . . . .	2231
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller . . .	2262
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller . . . . .	2688
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller . . .	2816

activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller . . . . .	2942
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller . . . . .	2324
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller . . . . .	3092
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller . . . . .	3403
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller . . . . .	3680
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller . . . . .	3827
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller . . . . .	2371
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller . . . . .	4048
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller . . . . .	4008
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller . . . . .	349
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller . . . . .	507
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller . . . . .	596
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller . . . . .	625
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller . . . . .	654
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller . . . . .	715
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller . . . . .	800
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller . . . . .	924
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller . . . . .	1292
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller . . . . .	1300
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller . . . . .	1361
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller . . . . .	1406
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller . . . . .	1468
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller . . . . .	1497
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller . . . . .	1732
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller . . . . .	1944
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller . . . . .	2273
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller . . . . .	2569
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller . . . . .	2606
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller . . . . .	2639
activemq::wireformat::openwire::marshal::v2::MessageMarshaller . . . . .	2703
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller . . . . .	221
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller . . . . .	269
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller . . . . .	390
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller . . . . .	417
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller . . . . .	462
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller . . . . .	568
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller . . . . .	687
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller . . . . .	2744
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller . . . . .	3040
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller . . . . .	3105
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller . . . . .	3198
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller . . . . .	3235
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller . . . . .	3262
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller . . . . .	3300
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller . . . . .	1531
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller . . . . .	1594
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller . . . . .	1846
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller . . . . .	2098
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller . . . . .	3427
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller . . . . .	3477
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller . . . . .	3871
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller . . . . .	892
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller . . . . .	1331

activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller . . . . .	1435
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller . . . . .	1766
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller . . . . .	2163
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller . . . . .	2192
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller . . . . .	2215
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller . . . . .	2246
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller . . . . .	2668
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller . . . . .	2796
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller . . . . .	2926
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller . . . . .	2312
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller . . . . .	3072
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller . . . . .	3383
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller . . . . .	3696
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller . . . . .	3831
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller . . . . .	2355
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller . . . . .	4040
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller . . . . .	4000
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller . . . . .	333
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller . . . . .	491
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller . . . . .	580
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller . . . . .	609
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller . . . . .	638
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller . . . . .	695
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller . . . . .	765
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller . . . . .	904
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller . . . . .	1272
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller . . . . .	1304
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller . . . . .	1365
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller . . . . .	1410
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller . . . . .	1472
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller . . . . .	1501
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller . . . . .	1736
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller . . . . .	1948
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller . . . . .	2277
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller . . . . .	2573
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller . . . . .	2610
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller . . . . .	2643
activemq::wireformat::openwire::marshal::v3::MessageMarshaller . . . . .	2698
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller . . . . .	209
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller . . . . .	249
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller . . . . .	374
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller . . . . .	401
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller . . . . .	446
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller . . . . .	552
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller . . . . .	667
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller . . . . .	2752
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller . . . . .	3048
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller . . . . .	3117
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller . . . . .	3206
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller . . . . .	3231
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller . . . . .	3266
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller . . . . .	3310
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller . . . . .	1535



activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller . . .	1598
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller . . .	1850
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller . . .	2102
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller . . . . .	3423
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller . . . . .	3489
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller . . . . .	3859
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller . . . . .	872
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller . . . . .	1335
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller . . . . .	1439
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller . . . . .	1770
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller . . . . .	2171
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller . . . . .	2196
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller . . . . .	2219
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller . . . . .	2250
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller . . . . .	2680
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller . . . . .	2808
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller . . . . .	2934
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller . . . . .	2308
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller . . . . .	3080
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller . . . . .	3399
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller . . . . .	3676
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller . . . . .	3835
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller . . . . .	2359
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller . . . . .	4052
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller . . . . .	4012
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller . . . . .	341
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller . . . . .	499
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller . . . . .	588
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller . . . . .	617
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller . . . . .	642
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller . . . . .	699
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller . . . . .	772
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller . . . . .	908
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller . . . . .	1276
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller . . . . .	1308
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller . . . . .	1369
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller . . . . .	1414
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller . . . . .	1476
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller . . . . .	1505
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller . . . . .	1740
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller . . . . .	1952
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller . . . . .	2281
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller . . . . .	2577
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller . . . . .	2618
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller . . . . .	2647
activemq::wireformat::openwire::marshal::v4::MessageMarshaller . . . . .	2708
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller . . . . .	217
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller . . . . .	257
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller . . . . .	382
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller . . . . .	409
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller . . . . .	454
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller . . . . .	560
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller . . . . .	671

activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller . . . . .	2756
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller . . . . .	3044
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller . . . . .	3101
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller . . . . .	3218
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	3247
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller . . .	3254
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller . . . . .	3295
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1539
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller . . .	1602
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1858
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller . .	2106
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller . . . . .	3431
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller . . . . .	3493
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller . . . .	3867
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller . . . . .	876
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller . . . . .	1339
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller . . . . .	1443
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller . . . . .	1774
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller . . . . .	2175
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller . . . . .	2204
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller . . . . .	2227
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller . . . .	2258
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller . . . . .	2672
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller . . .	2812
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller . . . . .	2938
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller .	2320
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller . . . . .	3076
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller . . . . .	3387
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller . . . . .	3688
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller . . . . .	3839
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller . .	2367
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller . . .	4044
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller . . . . .	4004
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller . . .	345
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller . . . .	503
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	592
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	621
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	650
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller . . . .	707
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller . . . . .	786
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller . . . . .	916
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller . .	1284
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller . . .	1316
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller . . . .	1377
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller . . .	1422
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller . . . . .	1484
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller . . .	1513
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller . . . .	1752
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller . . . .	1960
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller . . . . .	2285
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller . . . . .	2585
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller . . .	2614
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2655

activemq::wireformat::openwire::marshal::v5::MessageMarshaller . . . . .	2693
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	225
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	261
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	386
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller . . .	413
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	458
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	564
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	679
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller . . . . .	2748
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller . . . . .	3052
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller . . . . .	3113
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller . . . . .	3214
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	3243
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller . . .	3274
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller . . . . .	3305
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1547
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller . . .	1586
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1854
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller . .	2114
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller . . . . .	3415
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller . . . .	3485
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller . . .	3851
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller . . . . .	884
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller . . . . .	1347
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller . . . . .	1451
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller . . . .	1782
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller . . .	2167
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller . . .	2188
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller . . . . .	2235
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller . .	2254
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller . . . . .	2676
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller .	2804
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller . . . .	2930
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller .	2316
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller . . . . .	3084
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller . . . . .	3395
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller . . .	3684
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller . . . . .	3823
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller . .	2363
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller . . .	4056
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller . . . .	3992
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller . .	353
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller . . . .	511
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	600
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	629
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	658
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller . . . .	711
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller . . . . .	793
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller . . . . .	920
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller . .	1288
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller . . .	1320
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller . . . .	1381
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller . . .	1426

activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller . . . . .	1488
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller . . . . .	1517
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller . . . . .	1748
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller . . . . .	1940
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller . . . . .	2269
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller . . . . .	2565
activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller . . . . .	2626
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	2635
activemq::wireformat::openwire::marshal::v6::MessageMarshaller . . . . .	2718
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	229
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	265
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	394
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller . . . . .	421
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	466
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	572
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	683
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller . . . . .	2764
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller . . . . .	3056
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller . . . . .	3121
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller . . . . .	3202
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	3239
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller . . . . .	3270
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller . . . . .	3320
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller	1551
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller . . . . .	1590
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1842
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller . . . . .	2094
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller . . . . .	3411
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller . . . . .	3473
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller . . . . .	3863
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller . . . . .	888
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller . . . . .	1351
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller . . . . .	1455
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller . . . . .	1762
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller . . . . .	2159
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller . . . . .	2200
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller . . . . .	2223
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller . . . . .	2242
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller . . . . .	2684
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller . . . . .	2800
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller . . . . .	2922
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller . . . . .	2304
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller . . . . .	3088
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller . . . . .	3391
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller . . . . .	3692
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller . . . . .	3843
activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller . . . . .	2351
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller . . . . .	4036
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller . . . . .	3996
decaf::internal::net::ssl::DefaultSSLContext . . . . .	1691
decaf::util::zip::Deflater . . . . .	1706
cms::DeliveryMode . . . . .	1721
cms::Destination . . . . .	1723

cms::Queue . . . . .	3148
activemq::commands::ActiveMQQueue . . . . .	483
cms::TemporaryQueue . . . . .	3759
activemq::commands::ActiveMQTempQueue . . . . .	604
cms::TemporaryTopic . . . . .	3761
activemq::commands::ActiveMQTempTopic . . . . .	633
cms::Topic . . . . .	3817
activemq::commands::ActiveMQTopic . . . . .	691
activemq::commands::ActiveMQDestination::DestinationFilter . . . . .	1726
activemq::cmsutil::DestinationResolver . . . . .	1756
activemq::cmsutil::DynamicDestinationResolver . . . . .	1821
activemq::core::DispatchData . . . . .	1786
activemq::core::Dispatcher . . . . .	1787
activemq::core::ActiveMQConsumer . . . . .	310
activemq::core::ActiveMQSession . . . . .	515
decaf::lang::DYNAMIC_CAST_TOKEN . . . . .	1820
decaf::util::Map< K, V, COMPARATOR >::Entry . . . . .	1824
decaf::util::logging::ErrorManager . . . . .	1828
cms::ExceptionListener . . . . .	1838
decaf::util::concurrent::Executor . . . . .	1869
decaf::util::concurrent::ExecutorService . . . . .	1871
decaf::io::FileDescriptor . . . . .	1891
decaf::internal::net::SocketFileDescriptor . . . . .	3528
decaf::util::logging::Filter . . . . .	1893
decaf::io::Flushable . . . . .	1936
decaf::io::OutputStream . . . . .	2907
decaf::io::Writer . . . . .	4021
decaf::util::logging::Formatter . . . . .	1964
decaf::util::logging::SimpleFormatter . . . . .	3500
decaf::util::logging::XMLFormatter . . . . .	4060
decaf::util::concurrent::Future< V > . . . . .	1966
activemq::transport::correlator::FutureResponse . . . . .	1969
gz_header_s . . . . .	1975
gz_state . . . . .	1976
decaf::internal::util::HexStringParser . . . . .	1982
activemq::wireformat::openwire::utils::HexTable . . . . .	1984
activemq::util::IdGenerator . . . . .	1989
decaf::net::InetAddress . . . . .	2016
decaf::net::Inet4Address . . . . .	2011
decaf::net::Inet6Address . . . . .	2015
inflate_state . . . . .	2024
decaf::util::zip::Inflater . . . . .	2027
internal_state . . . . .	2118
decaf::lang::Iterable< E > . . . . .	2152
decaf::util::Collection< E > . . . . .	1184
decaf::util::AbstractCollection< E > . . . . .	179
decaf::util::List< E > . . . . .	2337
decaf::util::AbstractList< E > . . . . .	192
decaf::util::AbstractSequentialList< E > . . . . .	198
decaf::util::StlList< E > . . . . .	3589

decaf::util::Queue< E > . . . . .	3149
decaf::util::AbstractQueue< E > . . . . .	194
decaf::util::concurrent::BlockingQueue< E > . . . . .	843
decaf::util::concurrent::SynchronousQueue< E > . . . . .	3716
decaf::util::PriorityQueue< E > . . . . .	3028
decaf::util::Set< E > . . . . .	3439
decaf::util::AbstractSet< E > . . . . .	199
decaf::util::StlSet< E > . . . . .	3623
decaf::lang::Iterable< PrimitiveValueNode > . . . . .	2152
decaf::util::Collection< PrimitiveValueNode > . . . . .	1184
decaf::util::AbstractCollection< PrimitiveValueNode > . . . . .	179
decaf::util::List< PrimitiveValueNode > . . . . .	2337
decaf::util::StlList< PrimitiveValueNode > . . . . .	3589
activemq::util::PrimitiveList . . . . .	2980
decaf::util::Iterator< T > . . . . .	2154
decaf::util::Iterator< E > . . . . .	2154
decaf::util::ListIterator< E > . . . . .	2344
decaf::security::Key . . . . .	2293
decaf::security::PublicKey . . . . .	3140
decaf::util::comparators::Less< E > . . . . .	2328
decaf::util::concurrent::Lock . . . . .	2375
decaf::util::concurrent::locks::Lock . . . . .	2377
decaf::util::concurrent::locks::ReentrantLock . . . . .	3182
decaf::util::concurrent::locks::LockSupport . . . . .	2383
decaf::util::logging::Logger . . . . .	2386
decaf::util::logging::LoggerHierarchy . . . . .	2398
decaf::util::logging::LogManager . . . . .	2406
decaf::util::logging::LogRecord . . . . .	2413
decaf::util::logging::LogWriter . . . . .	2418
activemq::util::LongSequenceGenerator . . . . .	2455
decaf::util::logging::MarkBlockLogger . . . . .	2483
activemq::wireformat::MarshalAware . . . . .	2484
activemq::commands::DataStructure . . . . .	1660
activemq::commands::BaseDataStructure . . . . .	830
activemq::commands::ActiveMQDestination . . . . .	322
activemq::commands::ActiveMQQueue . . . . .	483
activemq::commands::ActiveMQTempDestination . . . . .	576
activemq::commands::ActiveMQTopic . . . . .	691
activemq::commands::BooleanExpression . . . . .	855
activemq::commands::BrokerId . . . . .	869
activemq::commands::Command . . . . .	1194
activemq::commands::BaseCommand . . . . .	758
activemq::commands::BrokerError . . . . .	863
activemq::commands::BrokerInfo . . . . .	896
activemq::commands::ConnectionControl . . . . .	1267
activemq::commands::ConnectionError . . . . .	1296
activemq::commands::ConnectionInfo . . . . .	1355
activemq::commands::ConsumerControl . . . . .	1401
activemq::commands::ConsumerInfo . . . . .	1459
activemq::commands::ControlCommand . . . . .	1493
activemq::commands::DestinationInfo . . . . .	1727

activemq::commands::FlushCommand . . . . .	1937
activemq::commands::KeepAliveInfo . . . . .	2266
activemq::commands::Message . . . . .	2516
activemq::commands::ActiveMQMessageTemplate< T > . . . . .	425
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage > . . . . .	425
activemq::commands::ActiveMQBytesMessage . . . . .	233
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage > . . . . .	425
activemq::commands::ActiveMQMapMessage . . . . .	360
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	425
activemq::commands::ActiveMQBlobMessage . . . . .	204
activemq::commands::ActiveMQMessage . . . . .	398
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage > . . . . .	425
activemq::commands::ActiveMQObjectMessage . . . . .	443
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage > . . . . .	425
activemq::commands::ActiveMQStreamMessage . . . . .	537
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage > . . . . .	425
activemq::commands::ActiveMQTextMessage . . . . .	662
activemq::commands::MessageAck . . . . .	2559
activemq::commands::MessageDispatch . . . . .	2594
activemq::commands::MessageDispatchNotification . . . . .	2630
activemq::commands::MessagePull . . . . .	2739
activemq::commands::ProducerAck . . . . .	3036
activemq::commands::ProducerInfo . . . . .	3096
activemq::commands::RemoveInfo . . . . .	3194
activemq::commands::RemoveSubscriptionInfo . . . . .	3222
activemq::commands::ReplayCommand . . . . .	3251
activemq::commands::Response . . . . .	3285
activemq::commands::DataArrayResponse . . . . .	1528
activemq::commands::DataResponse . . . . .	1583
activemq::commands::ExceptionResponse . . . . .	1839
activemq::commands::IntegerResponse . . . . .	2091
activemq::state::Tracked . . . . .	3818
activemq::commands::SessionInfo . . . . .	3407
activemq::commands::ShutdownInfo . . . . .	3470
activemq::commands::TransactionInfo . . . . .	3847
activemq::commands::WireFormatInfo . . . . .	3982
activemq::commands::ConnectionId . . . . .	1327
activemq::commands::ConsumerId . . . . .	1430
activemq::commands::DiscoveryEvent . . . . .	1758
activemq::commands::JournalQueueAck . . . . .	2156
activemq::commands::JournalTopicAck . . . . .	2183
activemq::commands::JournalTrace . . . . .	2212
activemq::commands::JournalTransaction . . . . .	2239
activemq::commands::MessageId . . . . .	2663
activemq::commands::NetworkBridgeFilter . . . . .	2793
activemq::commands::PartialCommand . . . . .	2918
activemq::commands::LastPartialCommand . . . . .	2301
activemq::commands::ProducerId . . . . .	3067
activemq::commands::SessionId . . . . .	3379
activemq::commands::SubscriptionInfo . . . . .	3671
activemq::commands::TransactionId . . . . .	3819

activemq::wireformat::openwire::marshal::v6::MarshallerFactory . . . . .	2487
activemq::wireformat::openwire::marshal::v3::MarshallerFactory . . . . .	2488
activemq::wireformat::openwire::marshal::v4::MarshallerFactory . . . . .	2489
activemq::wireformat::openwire::marshal::v5::MarshallerFactory . . . . .	2490
activemq::wireformat::openwire::marshal::v1::MarshallerFactory . . . . .	2491
activemq::wireformat::openwire::marshal::v2::MarshallerFactory . . . . .	2492
activemq::util::MarshallingSupport . . . . .	2493
decaf::lang::Math . . . . .	2497
cms::Message . . . . .	2534
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	425
cms::BytesMessage . . . . .	1056
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage > . . . . .	425
cms::MapMessage . . . . .	2472
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage > . . . . .	425
cms::ObjectMessage . . . . .	2841
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage > . . . . .	425
cms::StreamMessage . . . . .	3652
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage > . . . . .	425
cms::TextMessage . . . . .	3763
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage > . . . . .	425
activemq::cmsutil::MessageCreator . . . . .	2593
cms::MessageEnumeration . . . . .	2659
activemq::core::ActiveMQQueueBrowser . . . . .	487
cms::MessageListener . . . . .	2692
activemq::wireformat::openwire::utils::MessagePropertyInterceptor . . . . .	2732
decaf::util::concurrent::MutexHandle . . . . .	2787
decaf::internal::util::concurrent::MutexImpl . . . . .	2788
decaf::internal::net::Network . . . . .	2790
decaf::lang::Number . . . . .	2835
decaf::lang::Byte . . . . .	960
decaf::lang::Character . . . . .	1100
decaf::lang::Double . . . . .	1788
decaf::lang::Float . . . . .	1904
decaf::lang::Integer . . . . .	2077
decaf::lang::Long . . . . .	2420
decaf::lang::Short . . . . .	3440
decaf::util::concurrent::atomic::AtomicInteger . . . . .	741
decaf::internal::net::ssl::openssl::OpenSSLParameters . . . . .	2845
decaf::lang::Pointer< T, REFCOUNTER > . . . . .	2946
decaf::util::concurrent::PooledThreadListener . . . . .	2971
decaf::util::concurrent::ThreadPool . . . . .	3777
activemq::core::PrefetchPolicy . . . . .	2976
activemq::core::policies::DefaultPrefetchPolicy . . . . .	1673
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller . . . . .	3001
activemq::util::PrimitiveValueNode::PrimitiveValue . . . . .	3008
activemq::util::PrimitiveValueConverter . . . . .	3010
activemq::util::PrimitiveValueNode . . . . .	3012
decaf::security::Principal . . . . .	3026
decaf::security::auth::x500::X500Principal . . . . .	4027
activemq::cmsutil::ProducerCallback . . . . .	3064
activemq::cmsutil::CmsTemplate::SendExecutor . . . . .	3350



activemq::state::ProducerState . . . . .	3125
decaf::util::Properties . . . . .	3126
decaf::util::logging::PropertiesChangeListener . . . . .	3135
decaf::util::Random . . . . .	3155
decaf::security::SecureRandom . . . . .	3331
decaf::lang::Readable . . . . .	3160
decaf::io::Reader . . . . .	3163
decaf::util::concurrent::locks::ReadWriteLock . . . . .	3172
activemq::core::RedeliveryPolicy . . . . .	3177
activemq::core::policies::DefaultRedeliveryPolicy . . . . .	1677
decaf::util::concurrent::RejectedExecutionHandler . . . . .	3192
decaf::internal::util::Resource . . . . .	3280
decaf::internal::util::GenericResource< T > . . . . .	1974
decaf::internal::util::ResourceLifecycleManager . . . . .	3281
activemq::cmsutil::ResourceLifecycleManager . . . . .	3282
activemq::transport::mock::ResponseBuilder . . . . .	3289
activemq::wireformat::openwire::OpenWireResponseBuilder . . . . .	2905
decaf::lang::Runnable . . . . .	3325
activemq::threads::CompositeTaskRunner . . . . .	1223
activemq::threads::DedicatedTaskRunner . . . . .	1671
activemq::transport::IOTransport . . . . .	2145
decaf::lang::Thread . . . . .	3765
activemq::transport::mock::InternalCommandListener . . . . .	2122
decaf::util::concurrent::PooledThread . . . . .	2968
decaf::util::TimerTask . . . . .	3801
activemq::transport::inactivity::ReadChecker . . . . .	3162
activemq::transport::inactivity::WriteChecker . . . . .	4020
decaf::lang::Runtime . . . . .	3326
decaf::internal::DecafRuntime . . . . .	1670
decaf::security::SecureRandomSpi . . . . .	3339
decaf::internal::security::SecureRandomImpl . . . . .	3336
decaf::internal::security::SecureRandomImpl . . . . .	3336
decaf::util::concurrent::Semaphore . . . . .	3341
decaf::net::ServerSocket . . . . .	3352
decaf::net::ssl::SSLServerSocket . . . . .	3554
decaf::internal::net::ssl::openssl::OpenSSLServerSocket . . . . .	2847
decaf::net::ServerSocketFactory . . . . .	3361
decaf::internal::net::DefaultServerSocketFactory . . . . .	1682
decaf::net::ssl::SSLServerSocketFactory . . . . .	3560
decaf::internal::net::ssl::DefaultSSLServerSocketFactory . . . . .	1692
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory . . . . .	2853
activemq::cmsutil::SessionCallback . . . . .	3378
activemq::cmsutil::CmsTemplate::ProducerExecutor . . . . .	3065
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor . . . . .	3278
activemq::cmsutil::CmsTemplate::ReceiveExecutor . . . . .	3174
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor . . . . .	3279
activemq::cmsutil::SessionPool . . . . .	3435
activemq::state::SessionState . . . . .	3437
decaf::util::logging::SimpleLogger . . . . .	3501
decaf::net::SocketAddress . . . . .	3519

decaf::net::InetSocketAddress . . . . .	2023
decaf::net::SocketError . . . . .	3520
decaf::net::SocketFactory . . . . .	3524
decaf::internal::net::DefaultSocketFactory . . . . .	1686
decaf::net::ssl::SSLSocketFactory . . . . .	3571
decaf::internal::net::ssl::DefaultSSLSocketFactory . . . . .	1697
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory . . . . .	2875
decaf::net::SocketImplFactory . . . . .	3537
decaf::net::SocketOptions . . . . .	3538
decaf::net::SocketImpl . . . . .	3529
decaf::internal::net::tcp::TcpSocket . . . . .	3738
decaf::net::ssl::SSLContext . . . . .	3545
decaf::net::ssl::SSLContextSpi . . . . .	3548
decaf::internal::net::ssl::openssl::OpenSSLContextSpi . . . . .	2842
decaf::net::ssl::SSLParameters . . . . .	3551
activemq::commands::BrokerError::StackTraceElement . . . . .	3578
cms::Startable . . . . .	3586
cms::Connection . . . . .	1262
decaf::lang::STATIC_CAST_TOKEN . . . . .	3587
activemq::core::ActiveMQConstants::StaticInitializer . . . . .	3588
activemq::wireformat::stomp::StompCommandConstants . . . . .	3629
activemq::wireformat::stomp::StompFrame . . . . .	3633
activemq::wireformat::stomp::StompHelper . . . . .	3638
cms::Stoppable . . . . .	3648
cms::Connection . . . . .	1262
decaf::util::StringTokenizer . . . . .	3668
decaf::util::concurrent::Synchronizable . . . . .	3700
activemq::core::MessageDispatchChannel . . . . .	2599
decaf::util::Collection< PrimitiveValueNode > . . . . .	1184
decaf::internal::util::concurrent::SynchronizableImpl . . . . .	3711
decaf::io::InputStream . . . . .	2043
decaf::io::OutputStream . . . . .	2907
decaf::util::Collection< E > . . . . .	1184
decaf::util::concurrent::Mutex . . . . .	2782
decaf::util::Map< K, V, COMPARATOR > . . . . .	2459
decaf::util::AbstractMap< K, V, COMPARATOR > . . . . .	193
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > . . . . .	1228
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > . . . . .	1233
decaf::util::StlMap< K, V, COMPARATOR > . . . . .	3602
decaf::util::StlQueue< T > . . . . .	3615
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string > > . . . . .	2459
decaf::util::StlMap< std::string, PrimitiveValueNode > . . . . .	3602
activemq::util::PrimitiveMap . . . . .	2991
activemq::core::Synchronization . . . . .	3715
decaf::lang::System . . . . .	3726
activemq::threads::Task . . . . .	3734
activemq::core::ActiveMQSessionExecutor . . . . .	533
activemq::threads::CompositeTask . . . . .	1221
activemq::transport::failover::BackupTransportPool . . . . .	755
activemq::transport::failover::CloseTransportsTask . . . . .	1151
activemq::transport::failover::FailoverTransport . . . . .	1873

activemq::threads::CompositeTaskRunner . . . . .	1223
decaf::util::concurrent::TaskListener . . . . .	3735
activemq::threads::TaskRunner . . . . .	3736
activemq::threads::CompositeTaskRunner . . . . .	1223
activemq::threads::DedicatedTaskRunner . . . . .	1671
decaf::util::concurrent::ThreadFactory . . . . .	3774
decaf::lang::ThreadGroup . . . . .	3776
decaf::lang::Throwable . . . . .	3783
decaf::lang::Exception . . . . .	1831
activemq::exceptions::ActiveMQException . . . . .	357
activemq::exceptions::BrokerException . . . . .	867
decaf::io::IOException . . . . .	2142
decaf::io::EOFException . . . . .	1825
decaf::io::InterruptedIOException . . . . .	2127
decaf::net::SocketTimeoutException . . . . .	3542
decaf::io::UnsupportedEncodingException . . . . .	3913
decaf::io::UTFDataFormatException . . . . .	3966
decaf::net::HttpRetryException . . . . .	1986
decaf::net::MalformedURLException . . . . .	2456
decaf::net::ProtocolException . . . . .	3137
decaf::net::SocketException . . . . .	3521
decaf::internal::net::ssl::openssl::OpenSSLSocketException . . . . .	2871
decaf::net::BindException . . . . .	836
decaf::net::ConnectException . . . . .	1259
decaf::net::NoRouteToHostException . . . . .	2820
decaf::net::PortUnreachableException . . . . .	2973
decaf::net::UnknownHostException . . . . .	3907
decaf::net::UnknownServiceException . . . . .	3910
decaf::util::zip::ZipException . . . . .	4064
decaf::lang::exceptions::ClassCastException . . . . .	1145
decaf::lang::exceptions::IllegalArgumentException . . . . .	1991
decaf::lang::exceptions::IllegalMonitorStateException . . . . .	1994
decaf::lang::exceptions::IllegalStateException . . . . .	1998
decaf::nio::InvalidMarkException . . . . .	2135
decaf::lang::exceptions::IllegalThreadStateException . . . . .	2001
decaf::lang::exceptions::IndexOutOfBoundsException . . . . .	2008
decaf::lang::exceptions::InterruptedException . . . . .	2124
decaf::lang::exceptions::InvalidStateException . . . . .	2139
decaf::lang::exceptions::NoSuchElementException . . . . .	2826
decaf::lang::exceptions::NullPointerException . . . . .	2832
decaf::lang::exceptions::NumberFormatException . . . . .	2838
decaf::lang::exceptions::RuntimeException . . . . .	3328
decaf::lang::exceptions::UnsupportedOperationException . . . . .	3916
decaf::nio::ReadOnlyBufferException . . . . .	3169
decaf::net::URISyntaxException . . . . .	3948
decaf::nio::BufferOverflowException . . . . .	954
decaf::nio::BufferUnderflowException . . . . .	957
decaf::security::GeneralSecurityException . . . . .	1971
decaf::security::cert::CertificateException . . . . .	1092
decaf::security::cert::CertificateEncodingException . . . . .	1090
decaf::security::cert::CertificateExpiredException . . . . .	1094
decaf::security::cert::CertificateNotYetValidException . . . . .	1096

decaf::security::cert::CertificateParsingException . . . . .	1098
decaf::security::KeyException . . . . .	2295
decaf::security::InvalidKeyException . . . . .	2132
decaf::security::KeyManagementException . . . . .	2298
decaf::security::NoSuchAlgorithmException . . . . .	2823
decaf::security::NoSuchProviderException . . . . .	2829
decaf::security::SignatureException . . . . .	3497
decaf::util::concurrent::BrokenBarrierException . . . . .	860
decaf::util::concurrent::CancellationException . . . . .	1083
decaf::util::concurrent::ExecutionException . . . . .	1866
decaf::util::concurrent::RejectedExecutionException . . . . .	3189
decaf::util::concurrent::TimeoutException . . . . .	3787
decaf::util::zip::DataFormatException . . . . .	1555
decaf::util::Timer . . . . .	3790
decaf::internal::util::TimerTaskHeap . . . . .	3804
activemq::state::TransactionState . . . . .	3875
decaf::internal::util::concurrent::Transferer< E > . . . . .	3877
decaf::internal::util::concurrent::TransferQueue< E > . . . . .	3878
decaf::internal::util::concurrent::TransferStack< E > . . . . .	3881
activemq::transport::TransportFactory . . . . .	3889
activemq::transport::AbstractTransportFactory . . . . .	201
activemq::transport::failover::FailoverTransportFactory . . . . .	1885
activemq::transport::mock::MockTransportFactory . . . . .	2779
activemq::transport::tcp::TcpTransportFactory . . . . .	3756
activemq::transport::tcp::SslTransportFactory . . . . .	3576
activemq::transport::TransportListener . . . . .	3900
activemq::core::ActiveMQConnection . . . . .	273
activemq::transport::DefaultTransportListener . . . . .	1704
activemq::transport::failover::BackupTransport . . . . .	752
activemq::transport::mock::InternalCommandListener . . . . .	2122
activemq::transport::failover::FailoverTransportListener . . . . .	1888
activemq::transport::TransportFilter . . . . .	3891
activemq::transport::TransportRegistry . . . . .	3902
tree_desc_s . . . . .	3905
decaf::lang::Thread::UncaughtExceptionHandler . . . . .	3906
decaf::internal::net::URIEncoderDecoder . . . . .	3932
decaf::internal::net::URIHelper . . . . .	3935
activemq::transport::failover::URIPool . . . . .	3942
activemq::util::URISupport . . . . .	3945
decaf::internal::net::URIType . . . . .	3952
decaf::net::URL . . . . .	3960
decaf::net::URLDecoder . . . . .	3962
decaf::net::URLEncoder . . . . .	3963
activemq::util::Usage . . . . .	3964
activemq::util::MemoryUsage . . . . .	2512
activemq::wireformat::WireFormat . . . . .	3976
activemq::wireformat::openwire::OpenWireFormat . . . . .	2887
activemq::wireformat::stomp::StompWireFormat . . . . .	3643
activemq::wireformat::WireFormatFactory . . . . .	3980
activemq::wireformat::openwire::OpenWireFormatFactory . . . . .	2899
activemq::wireformat::stomp::StompWireFormatFactory . . . . .	3647

---

activemq::wireformat::WireFormatRegistry . . . . .	4017
z_stream_s . . . . .	4062



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<b>decaf::util::AbstractCollection</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Collection</b> (p.1184) interface, to minimize the effort required to implement this interface ) . . . . .	179
<b>decaf::util::AbstractList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p.2337) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array) ) . . . . .	192
<b>decaf::util::AbstractMap</b> < <b>K</b> , <b>V</b> , <b>COMPARATOR</b> > (This class provides a skeletal implementation of the <b>Map</b> (p.2459) interface, to minimize the effort required to implement this interface ) . . . . .	193
<b>decaf::util::AbstractQueue</b> < <b>E</b> > (This class provides skeletal implementations of some <b>Queue</b> (p.3149) operations ) . . . . .	194
<b>decaf::util::AbstractSequentialList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p.2337) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list) )	198
<b>decaf::util::AbstractSet</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Set</b> (p.3439) interface to minimize the effort required to implement this interface )	199
<b>activemq::transport::AbstractTransportFactory</b> (Abstract implementation of the <b>TransportFactory</b> (p.3889) interface, providing the base functionality that's common to most of the <b>TransportFactory</b> (p.3889) instances ) . . . . .	201
<b>activemq::core::ActiveMQAckHandler</b> (Interface class that is used to give CMS Messages an interface to Ack themselves with ) . . . . .	203
<b>activemq::commands::ActiveMQBlobMessage</b> . . . . .	204
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.209) ) . . . . .	209
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.213) ) . . . . .	213
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.217) ) . . . . .	217

<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.221) ) . . . . .	221
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.225) ) . . . . .	225
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p.229) ) . . . . .	229
<b>activemq::commands::ActiveMQBytesMessage</b> . . . . .	233
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.249) ) . . . . .	249
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.253) ) . . . . .	253
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.257) ) . . . . .	257
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.261) ) . . . . .	261
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.265) ) . . . . .	265
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p.269) ) . . . . .	269
<b>activemq::core::ActiveMQConnection</b> (Concrete connection used for all connectors to the ActiveMQ broker ) . . . . .	273
<b>activemq::core::ActiveMQConnectionFactory</b> . . . . .	292
<b>activemq::core::ActiveMQConnectionMetaData</b> (This class houses all the various settings and information that is used by an instance of an <b>ActiveMQConnection</b> (p.273) class ) . . . . .	303
<b>activemq::core::ActiveMQConstants</b> (Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values ) . . . . .	307
<b>activemq::core::ActiveMQConsumer</b> . . . . .	310
<b>activemq::library::ActiveMQCPP</b> . . . . .	320
<b>activemq::commands::ActiveMQDestination</b> . . . . .	322
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.333) ) . . . . .	333
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.337) ) . . . . .	337
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.341) ) . . . . .	341
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.345) ) . . . . .	345



<b>activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.349) ) . . . . .	349
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p.353) ) . . . . .	353
<b>activemq::exceptions::ActiveMQException</b> . . . . .	357
<b>activemq::commands::ActiveMQMapMessage</b> . . . . .	360
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.374) ) . . . . .	374
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.378) ) . . . . .	378
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.382) ) . . . . .	382
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.386) ) . . . . .	386
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.390) ) . . . . .	390
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p.394) ) . . . . .	394
<b>activemq::commands::ActiveMQMessage</b> . . . . .	398
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.401) ) . . . . .	401
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.405) ) . . . . .	405
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.409) ) . . . . .	409
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.413) ) . . . . .	413
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.417) ) . . . . .	417
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p.421) ) . . . . .	421
<b>activemq::commands::ActiveMQMessageTemplate&lt; T &gt;</b> . . . . .	425
<b>activemq::commands::ActiveMQObjectMessage</b> . . . . .	443
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p.446) ) . . . . .	446
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p.1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p.450) ) . . . . .	450

<b>activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 454) )	454
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 458) )	458
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 462) )	462
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 466) )	466
<b>activemq::core::ActiveMQProducer</b>	470
<b>activemq::util::ActiveMQProperties</b> (Implementation of the <b>CMSProperties</b> interface that delegates to a <b>decaf::util::Properties</b> (p. 3126) object )	478
<b>activemq::commands::ActiveMQQueue</b>	483
<b>activemq::core::ActiveMQQueueBrowser</b>	487
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 491) )	491
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 495) )	495
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 499) )	499
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 503) )	503
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 507) )	507
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 511) )	511
<b>activemq::core::ActiveMQSession</b>	515
<b>activemq::core::ActiveMQSessionExecutor</b> (Delegate dispatcher for a single session )	533
<b>activemq::commands::ActiveMQStreamMessage</b>	537
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 552) )	552
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 556) )	556
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 560) )	560
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 564) )	564
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 568) )	568

<b>activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 572) )	572
<b>activemq::commands::ActiveMQTempDestination</b>	576
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 580) )	580
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 584) )	584
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 588) )	588
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 592) )	592
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 596) )	596
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 600) )	600
<b>activemq::commands::ActiveMQTempQueue</b>	604
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 609) )	609
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 613) )	613
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 617) )	617
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 621) )	621
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 625) )	625
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 629) )	629
<b>activemq::commands::ActiveMQTempTopic</b>	633
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 638) )	638
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 642) )	642
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 646) )	646
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 650) )	650

<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 654) ) . . . . .	654
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 658) ) . . . . .	658
<b>activemq::commands::ActiveMQTextMessage</b> . . . . .	662
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 667) ) . . . . .	667
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 671) ) . . . . .	671
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 675) ) . . . . .	675
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 679) ) . . . . .	679
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 683) ) . . . . .	683
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 687) ) . . . . .	687
<b>activemq::commands::ActiveMQTopic</b> . . . . .	691
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 695) ) . . . . .	695
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 699) ) . . . . .	699
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 703) ) . . . . .	703
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 707) ) . . . . .	707
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 711) ) . . . . .	711
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 715) ) . . . . .	715
<b>activemq::core::ActiveMQTransactionContext</b> (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back) . . . . .	719
<b>decaf::util::zip::Adler32</b> (Clas that can be used to compute an Adler-32 Checksum (p. 1142) for a data stream) . . . . .	722
<b>decaf::lang::Appendable</b> (An object to which char sequences and values can be appended) . . . . .	725
<b>decaf::internal::AprPool</b> (Wraps an APR pool object so that classes in <b>decaf</b> (p. 143) can create a static member for use in static methods where apr function calls that need a pool are made) . . . . .	728

<b>decaf::lang::ArrayPointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p. 2946) that is a template on a Type and is <b>Thread</b> (p. 3765) Safe if the default Reference Counter is used ) . . . . .	730
<b>decaf::lang::ArrayPointerComparator&lt; T, R &gt;</b> (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this <b>ArrayPointer</b> (p. 730) ) . . . . .	737
<b>decaf::util::concurrent::atomic::AtomicBoolean</b> (A boolean value that may be updated atomically ) . . . . .	738
<b>decaf::util::concurrent::atomic::AtomicInteger</b> (An int value that may be updated atomically ) . . . . .	741
<b>decaf::util::concurrent::atomic::AtomicRefCounter</b> . . . . .	746
<b>decaf::util::concurrent::atomic::AtomicReference&lt; T &gt;</b> (An <b>Pointer</b> reference that may be updated atomically ) . . . . .	749
<b>activemq::transport::failover::BackupTransport</b> . . . . .	752
<b>activemq::transport::failover::BackupTransportPool</b> . . . . .	755
<b>activemq::commands::BaseCommand</b> . . . . .	758
<b>activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 765) ) . . . . .	765
<b>activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 772) ) . . . . .	772
<b>activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 779) ) . . . . .	779
<b>activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 786) ) . . . . .	786
<b>activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 793) ) . . . . .	793
<b>activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 800) ) . . . . .	800
<b>activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller</b> (Base class for all Marshallers that <b>marshal</b> (p. 107) <b>DataStructures</b> to and from the wire using the OpenWire protocol ) . . . . .	807
<b>activemq::commands::BaseDataStructure</b> . . . . .	830
<b>binary_function</b> . . . . .	835
<b>decaf::net::BindException</b> . . . . .	836
<b>decaf::io::BlockingByteArrayInputStream</b> (This is a blocking version of a byte buffer stream ) . . . . .	839
<b>decaf::util::concurrent::BlockingQueue&lt; E &gt;</b> (A <b>decaf::util::Queue</b> (p. 3149) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element ) . . . . .	843
<b>decaf::lang::Boolean</b> . . . . .	850
<b>activemq::commands::BooleanExpression</b> . . . . .	855
<b>activemq::wireformat::openwire::utils::BooleanStream</b> (Manages the writing and reading of boolean data streams to and from a data source such as a <b>DataInputStream</b> or <b>DataOutputStream</b> ) . . . . .	857
<b>decaf::util::concurrent::BrokenBarrierException</b> . . . . .	860

<b>activemq::commands::BrokerError</b> (This class represents an Exception sent from the Broker ) . . . . .	863
<b>activemq::exceptions::BrokerException</b> . . . . .	867
<b>activemq::commands::BrokerId</b> . . . . .	869
<b>activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 872) ) . . .	872
<b>activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 876) ) . . .	876
<b>activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 880) ) . . .	880
<b>activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 884) ) . . .	884
<b>activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 888) ) . . .	888
<b>activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 892) ) . . .	892
<b>activemq::commands::BrokerInfo</b> . . . . .	896
<b>activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 904) ) . . . . .	904
<b>activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 908) ) . . . . .	908
<b>activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 912) ) . . . . .	912
<b>activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 916) ) . . . . .	916
<b>activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 920) ) . . . . .	920
<b>activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 924) ) . . . . .	924
<b>decaf::nio::Buffer</b> (A container for data of a specific primitive type ) . . . . .	928
<b>decaf::io::BufferedInputStream</b> (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of <b>io</b> (p. 154) operations on the input stream ) . . . . .	934
<b>decaf::io::BufferedOutputStream</b> (Wrapper around another output stream that buffers output before writing to the target output stream ) . . . . .	940
<b>decaf::internal::nio::BufferFactory</b> (Factory class used by static methods in the <b>decaf::nio</b> (p. 163) package to create the various default version of the NIO interfaces ) . . . . .	942
<b>decaf::nio::BufferOverflowException</b> . . . . .	954
<b>decaf::nio::BufferUnderflowException</b> . . . . .	957
<b>decaf::lang::Byte</b> . . . . .	960
<b>decaf::internal::util::ByteArrayAdapter</b> (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data ) . . . . .	969
<b>decaf::internal::nio::ByteArrayBuffer</b> (This class defines six categories of operations upon byte buffers: ) . . . . .	991
<b>decaf::io::ByteArrayInputStream</b> (A <b>ByteArrayInputStream</b> (p. 1020) contains an <b>internal</b> (p. 144) buffer that contains bytes that may be read from the stream ) 1020	

<b>decaf::io::ByteArrayOutputStream</b> . . . . .	1028
<b>decaf::nio::ByteBuffer</b> (This class defines six categories of operations upon byte buffers: ) . . . . .	1031
<b>cms::BytesMessage</b> (A <b>BytesMessage</b> (p.1056) object is used to send a message containing a stream of unsigned bytes ) . . . . .	1056
<b>activemq::cmsutil::CachedConsumer</b> (A cached message consumer contained within a pooled session ) . . . . .	1071
<b>activemq::cmsutil::CachedProducer</b> (A cached message producer contained within a pooled session ) . . . . .	1075
<b>decaf::util::concurrent::Callable&lt; V &gt;</b> (A task that returns a result and may throw an exception ) . . . . .	1082
<b>decaf::util::concurrent::CancellationException</b> . . . . .	1083
<b>decaf::security::cert::Certificate</b> (Base interface for all identity certificates ) . . . . .	1086
<b>decaf::security::cert::CertificateEncodingException</b> . . . . .	1090
<b>decaf::security::cert::CertificateException</b> . . . . .	1092
<b>decaf::security::cert::CertificateExpiredException</b> . . . . .	1094
<b>decaf::security::cert::CertificateNotYetValidException</b> . . . . .	1096
<b>decaf::security::cert::CertificateParsingException</b> . . . . .	1098
<b>decaf::lang::Character</b> . . . . .	1100
<b>decaf::internal::nio::CharArrayBuffer</b> . . . . .	1108
<b>decaf::nio::CharBuffer</b> (This class defines four categories of operations upon character buffers: ) . . . . .	1119
<b>decaf::lang::CharSequence</b> (A <b>CharSequence</b> (p. 1135) is a readable sequence of char values ) . . . . .	1135
<b>decaf::util::zip::CheckedInputStream</b> (An implementation of a <b>FilterInputStream</b> that will maintain a <b>Checksum</b> (p.1142) of the bytes read, the <b>Checksum</b> (p.1142) can then be used to verify the integrity of the input stream ) . . . . .	1137
<b>decaf::util::zip::CheckedOutputStream</b> (An implementation of a <b>FilterOutputStream</b> that will maintain a <b>Checksum</b> (p.1142) of the bytes written, the <b>Checksum</b> (p.1142) can then be used to verify the integrity of the output stream ) . . . . .	1140
<b>decaf::util::zip::Checksum</b> (An interface used to represent <b>Checksum</b> (p.1142) values in the Zip package ) . . . . .	1142
<b>decaf::lang::exceptions::ClassCastException</b> . . . . .	1145
<b>cms::Closeable</b> (Interface for a class that implements the close method ) . . . . .	1148
<b>decaf::io::Closeable</b> (Interface for a class that implements the close method ) . . . . .	1149
<b>activemq::transport::failover::CloseTransportsTask</b> . . . . .	1151
<b>activemq::cmsutil::CmsAccessor</b> (Base class for <b>activemq.cmsutil.CmsTemplate</b> (p.1170) and other CMS-accessing gateway helpers, defining common properties such as the CMS <b>cms.ConnectionFactory</b> (p.1324) to operate on ) . . . . .	1153
<b>activemq::cmsutil::CmsDestinationAccessor</b> (Extends the <b>CmsAccessor</b> (p.1153) to add support for resolving destination names ) . . . . .	1157
<b>cms::CMSException</b> (CMS API Exception that is the base for all exceptions thrown from CMS classes ) . . . . .	1160
<b>activemq::util::CMSExceptionSupport</b> . . . . .	1163
<b>cms::CMSProperties</b> (Interface for a Java-like properties object ) . . . . .	1165
<b>cms::CMSSecurityException</b> (This exception must be thrown when a provider rejects a user name/password submitted by a client ) . . . . .	1169
<b>activemq::cmsutil::CmsTemplate</b> ( <b>CmsTemplate</b> (p.1170) simplifies performing synchronous CMS operations ) . . . . .	1170
<b>code</b> . . . . .	1183
<b>decaf::util::Collection&lt; E &gt;</b> (The root interface in the collection hierarchy ) . . . . .	1184
<b>activemq::commands::Command</b> . . . . .	1194

<b>activemq::state::CommandVisitor</b> (Interface for an Object that can visit the various Command Objects that are sent from and to this client ) . . . . .	1200
<b>activemq::state::CommandVisitorAdapter</b> (Default Implementation of a <b>CommandVisitor</b> (p. 1200) that returns NULL for all calls ) . . . . .	1208
<b>decaf::lang::Comparable&lt; T &gt;</b> (This interface imposes a total ordering on the objects of each class that implements it ) . . . . .	1214
<b>decaf::util::Comparator&lt; T &gt;</b> (A comparison function, which imposes a total ordering on some collection of objects ) . . . . .	1217
<b>activemq::util::CompositeData</b> (Represents a Composite URI ) . . . . .	1219
<b>activemq::threads::CompositeTask</b> (Represents a single task that can be part of a set of Tasks that are contained in a <b>CompositeTaskRunner</b> (p. 1223) ) . . . .	1221
<b>activemq::threads::CompositeTaskRunner</b> (A <b>Task</b> (p. 3734) Runner that can contain one or more <b>CompositeTasks</b> that are each checked for pending work and run if any is present in the order that the tasks were added ) . . . . .	1223
<b>activemq::transport::CompositeTransport</b> (A <b>Composite Transport</b> (p. 3883) is a <b>Transport</b> (p. 3883) implementation that is composed of several <b>Transports</b> ) .	1226
<b>decaf::util::concurrent::ConcurrentMap&lt; K, V, COMPARATOR &gt;</b> (Interface for a <b>Map</b> (p. 2459) type that provides additional <b>atomic</b> (p. 173) <b>putIfAbsent</b> , <b>remove</b> , and <b>replace</b> methods alongside the already available <b>Map</b> (p. 2459) interface ) . . . . .	1228
<b>decaf::util::concurrent::ConcurrentStlMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2459) template that wraps around a <b>std::map</b> to provide a more user-friendly interface and to provide common functions that do not exist in <b>std::map</b> ) . . .	1233
<b>decaf::util::concurrent::locks::Condition</b> ( <b>Condition</b> (p. 1249) factors out the <b>Mutex</b> (p. 2782) monitor methods ( <b>wait</b> , <b>notify</b> and <b>notifyAll</b> ) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary <b>Lock</b> (p. 2377) implementations ) . . . . .	1249
<b>decaf::util::concurrent::ConditionHandle</b> . . . . .	1255
<b>decaf::internal::util::concurrent::ConditionImpl</b> . . . . .	1257
<b>decaf::net::ConnectException</b> . . . . .	1259
<b>cms::Connection</b> (The client's connection to its provider ) . . . . .	1262
<b>activemq::commands::ConnectionControl</b> . . . . .	1267
<b>activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1272) ) . . . . .	1272
<b>activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1276) ) . . . . .	1276
<b>activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1280) ) . . . . .	1280
<b>activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1284) ) . . . . .	1284
<b>activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1288) ) . . . . .	1288
<b>activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1292) ) . . . . .	1292
<b>activemq::commands::ConnectionError</b> . . . . .	1296
<b>activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1300) ) . . . . .	1300



<b>activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1304) )	1304
<b>activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1308) )	1308
<b>activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1312) )	1312
<b>activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1316) )	1316
<b>activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1320) )	1320
<b>cms::ConnectionFactory</b> (Defines the interface for a factory that creates connection objects, the <b>Connection</b> (p. 1262) objects returned implement the CMS <b>Connection</b> (p. 1262) interface and hide the CMS Provider specific implementation details behind that interface )	1324
<b>activemq::commands::ConnectionId</b>	1327
<b>activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1331) )	1331
<b>activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1335) )	1335
<b>activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1339) )	1339
<b>activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1343) )	1343
<b>activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1347) )	1347
<b>activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1351) )	1351
<b>activemq::commands::ConnectionInfo</b>	1355
<b>activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1361) )	1361
<b>activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1365) )	1365
<b>activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1369) )	1369
<b>activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1373) )	1373
<b>activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1377) )	1377

<b>activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p.1381) ) . . . . .	1381
<b>cms::ConnectionMetaData</b> (A <b>ConnectionMetaData</b> (p.1385) object provides information describing the <b>Connection</b> (p.1262) object ) . . . . .	1385
<b>activemq::state::ConnectionState</b> . . . . .	1389
<b>activemq::state::ConnectionStateTracker</b> . . . . .	1392
<b>decaf::util::logging::ConsoleHandler</b> (This <b>Handler</b> (p.1978) publishes log records to <b>System.err</b> ) . . . . .	1399
<b>activemq::commands::ConsumerControl</b> . . . . .	1401
<b>activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1406) ) . . . . .	1406
<b>activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1410) ) . . . . .	1410
<b>activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1414) ) . . . . .	1414
<b>activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1418) ) . . . . .	1418
<b>activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1422) ) . . . . .	1422
<b>activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerControlMarshaller</b> (p.1426) ) . . . . .	1426
<b>activemq::commands::ConsumerId</b> . . . . .	1430
<b>activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1435) ) . . . . .	1435
<b>activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1439) ) . . . . .	1439
<b>activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1443) ) . . . . .	1443
<b>activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1447) ) . . . . .	1447
<b>activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1451) ) . . . . .	1451
<b>activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerIdMarshaller</b> (p.1455) ) . . . . .	1455
<b>activemq::commands::ConsumerInfo</b> . . . . .	1459
<b>activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p.1468) ) . . . . .	1468
<b>activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p.1472) ) . . . . .	1472

<b>activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1476) ) . . . . .	1476
<b>activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1480) ) . . . . .	1480
<b>activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1484) ) . . . . .	1484
<b>activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1488) ) . . . . .	1488
<b>activemq::state::ConsumerState</b> . . . . .	1492
<b>activemq::commands::ControlCommand</b> . . . . .	1493
<b>activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1497) ) . . . . .	1497
<b>activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1501) ) . . . . .	1501
<b>activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1505) ) . . . . .	1505
<b>activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1509) ) . . . . .	1509
<b>activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1513) ) . . . . .	1513
<b>activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1517) ) . . . . .	1517
<b>decaf::util::concurrent::CountDownLatch</b> . . . . .	1521
<b>decaf::util::zip::CRC32</b> (Class that can be used to compute a CRC-32 checksum for a data stream ) . . . . .	1524
<b>ct_data_s</b> . . . . .	1527
<b>activemq::commands::DataArrayResponse</b> . . . . .	1528
<b>activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1531) ) . . . . .	1531
<b>activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1535) ) . . . . .	1535
<b>activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1539) ) . . . . .	1539
<b>activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1543) ) . . . . .	1543
<b>activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1547) ) . . . . .	1547

<b>activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1551) ) . . . . .	1551
<b>decaf::util::zip::DataFormatException</b> . . . . .	1555
<b>decaf::io::DataInput</b> (The <b>DataInput</b> (p. 1558) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types ) . . . . .	1558
<b>decaf::io::DataInputStream</b> (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way )	1566
<b>decaf::io::DataOutput</b> (The <b>DataOutput</b> (p. 1574) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream ) . . . . .	1574
<b>decaf::io::DataOutputStream</b> (A data output stream lets an application write primitive Java data types to an output stream in a portable way ) . . . . .	1579
<b>activemq::commands::DataResponse</b> . . . . .	1583
<b>activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1586) ) . . . . .	1586
<b>activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1590) ) . . . . .	1590
<b>activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1594) ) . . . . .	1594
<b>activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1598) ) . . . . .	1598
<b>activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1602) ) . . . . .	1602
<b>activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1606) ) . . . . .	1606
<b>activemq::wireformat::openwire::marshal::DataStreamMarshaller</b> (Base class for all classes that <b>marshal</b> (p. 107) <b>commands</b> (p. 87) for Openwire ) . . . .	1610
<b>activemq::commands::DataStructure</b> . . . . .	1660
<b>decaf::util::Date</b> (Wrapper class around a time value in milliseconds ) . . . . .	1665
<b>decaf::internal::DecafRuntime</b> (Handles APR initialization and termination ) . . . .	1670
<b>activemq::threads::DedicatedTaskRunner</b> . . . . .	1671
<b>activemq::core::policies::DefaultPrefetchPolicy</b> . . . . .	1673
<b>activemq::core::policies::DefaultRedeliveryPolicy</b> . . . . .	1677
<b>decaf::internal::net::DefaultServerSocketFactory</b> (Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options )	1682
<b>decaf::internal::net::DefaultSocketFactory</b> (SocketFactory implementation that is used to create Sockets ) . . . . .	1686
<b>decaf::internal::net::ssl::DefaultSSLContext</b> (Default SSLContext manager for the Decaf library ) . . . . .	1691
<b>decaf::internal::net::ssl::DefaultSSLServerSocketFactory</b> (Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1692

<b>decaf::internal::net::ssl::DefaultSSLSocketFactory</b> (Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1697
<b>activemq::transport::DefaultTransportListener</b> . . . . .	1704
<b>decaf::util::zip::Deflater</b> (This class compresses data using the <i>DEFLATE</i> algorithm (see specification) ) . . . . .	1706
<b>decaf::util::zip::DeflaterOutputStream</b> (Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream ) . .	1716
<b>decaf::util::concurrent::Delayed</b> (A mix-in style interface for marking objects that should be acted upon after a given delay ) . . . . .	1720
<b>cms::DeliveryMode</b> (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages ) . . . . .	1721
<b>cms::Destination</b> (A <b>Destination</b> (p. 1723) object encapsulates a provider-specific address ) . . . . .	1723
<b>activemq::commands::ActiveMQDestination::DestinationFilter</b> . . . . .	1726
<b>activemq::commands::DestinationInfo</b> . . . . .	1727
<b>activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1732) ) . . . . .	1732
<b>activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1736) ) . . . . .	1736
<b>activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1740) ) . . . . .	1740
<b>activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1744) ) . . . . .	1744
<b>activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1748) ) . . . . .	1748
<b>activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1752) ) . . . . .	1752
<b>activemq::cmsutil::DestinationResolver</b> (Resolves a CMS destination name to a <b>Destination</b> ) . . . . .	1756
<b>activemq::commands::DiscoveryEvent</b> . . . . .	1758
<b>activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1762) ) . . . . .	1762
<b>activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1766) ) . . . . .	1766
<b>activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1770) ) . . . . .	1770
<b>activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1774) ) . . . . .	1774
<b>activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1778) ) . . . . .	1778

<b>activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p.1782) ) . . . . .	1782
<b>activemq::core::DispatchData</b> (Simple POJO that contains the information necessary to route a message to a specified consumer ) . . . . .	1786
<b>activemq::core::Dispatcher</b> (Interface for an object responsible for dispatching messages to consumers ) . . . . .	1787
<b>decaf::lang::Double</b> . . . . .	1788
<b>decaf::internal::nio::DoubleArrayBuffer</b> . . . . .	1799
<b>decaf::nio::DoubleBuffer</b> (This class defines four categories of operations upon double buffers: ) . . . . .	1809
<b>decaf::lang::DYNAMIC_CAST_TOKEN</b> . . . . .	1820
<b>activemq::cmsutil::DynamicDestinationResolver</b> (Resolves a CMS destination name to a <b>Destination</b> ) . . . . .	1821
<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;::Entry</b> . . . . .	1824
<b>decaf::io::EOFException</b> . . . . .	1825
<b>decaf::util::logging::ErrorManager</b> ( <b>ErrorManager</b> (p.1828) objects can be attached to <b>Handlers</b> to process any error that occur on a <b>Handler</b> (p.1978) during Logging ) . . . . .	1828
<b>decaf::lang::Exception</b> . . . . .	1831
<b>cms::ExceptionListener</b> (If a CMS provider detects a serious problem, it notifies the client application through an <b>ExceptionListener</b> (p.1838) that is registered with the <b>Connection</b> (p.1262) ) . . . . .	1838
<b>activemq::commands::ExceptionResponse</b> . . . . .	1839
<b>activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1842) ) . . . . .	1842
<b>activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1846) ) . . . . .	1846
<b>activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1850) ) . . . . .	1850
<b>activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1854) ) . . . . .	1854
<b>activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1858) ) . . . . .	1858
<b>activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p.1862) ) . . . . .	1862
<b>decaf::util::concurrent::ExecutionException</b> . . . . .	1866
<b>decaf::util::concurrent::Executor</b> (An object that executes submitted <b>decaf.lang Runnable</b> (p.3325) tasks ) . . . . .	1869
<b>decaf::util::concurrent::ExecutorService</b> (An <b>Executor</b> (p.1869) that provides methods to manage termination and methods that can produce a <b>Future</b> (p.1966) for tracking progress of one or more asynchronous tasks ) . . . . .	1871
<b>activemq::transport::failover::FailoverTransport</b> . . . . .	1873
<b>activemq::transport::failover::FailoverTransportFactory</b> (Creates an instance of a <b>FailoverTransport</b> (p.1873) ) . . . . .	1885
<b>activemq::transport::failover::FailoverTransportListener</b> (Utility class used by the <b>Transport</b> (p.3883) to perform the work of responding to events from the active <b>Transport</b> (p.3883) ) . . . . .	1888

<b>decaf::io::FileDescriptor</b> (This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files ) . . . . .	1891
<b>decaf::util::logging::Filter</b> (A <b>Filter</b> (p. 1893) can be used to provide fine grain control over what is logged, beyond the control provided by log levels ) . . . . .	1893
<b>decaf::io::FilterInputStream</b> (A <b>FilterInputStream</b> (p. 1894) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality ) . . . . .	1894
<b>decaf::io::FilterOutputStream</b> (This class is the superclass of all classes that filter output streams ) . . . . .	1900
<b>decaf::lang::Float</b> . . . . .	1904
<b>decaf::internal::nio::FloatArrayBuffer</b> . . . . .	1915
<b>decaf::nio::FloatBuffer</b> (This class defines four categories of operations upon float buffers: ) . . . . .	1925
<b>decaf::io::Flushable</b> (A <b>Flushable</b> (p. 1936) is a destination of data that can be flushed )	1936
<b>activemq::commands::FlushCommand</b> . . . . .	1937
<b>activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1940) ) . . . . .	1940
<b>activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1944) ) . . . . .	1944
<b>activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1948) ) . . . . .	1948
<b>activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1952) ) . . . . .	1952
<b>activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1956) ) . . . . .	1956
<b>activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1960) ) . . . . .	1960
<b>decaf::util::logging::Formatter</b> (A <b>Formatter</b> (p. 1964) provides support for formatting LogRecords ) . . . . .	1964
<b>decaf::util::concurrent::Future&lt; V &gt;</b> (A <b>Future</b> (p. 1966) represents the result of an asynchronous computation ) . . . . .	1966
<b>activemq::transport::correlator::FutureResponse</b> (A container that holds a response object ) . . . . .	1969
<b>decaf::security::GeneralSecurityException</b> . . . . .	1971
<b>decaf::internal::util::GenericResource&lt; T &gt;</b> (A <b>Generic Resource</b> (p. 3280) wraps some type and will delete it when the <b>Resource</b> (p. 3280) itself is deleted ) . .	1974
<b>gz_header_s</b> . . . . .	1975
<b>gz_state</b> . . . . .	1976
<b>decaf::util::logging::Handler</b> (A <b>Handler</b> (p. 1978) object takes log messages from a <b>Logger</b> (p. 2386) and exports them ) . . . . .	1978
<b>decaf::internal::util::HexStringParser</b> . . . . .	1982
<b>activemq::wireformat::openwire::utils::HexTable</b> (Maps hexadecimal strings to the value of an index into the table, i.e ) . . . . .	1984
<b>decaf::net::HttpRetryException</b> . . . . .	1986
<b>activemq::util::IdGenerator</b> . . . . .	1989
<b>decaf::lang::exceptions::IllegalArgumentException</b> . . . . .	1991
<b>decaf::lang::exceptions::IllegalMonitorStateException</b> . . . . .	1994

<b>cms::IllegalStateException</b> (This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation )	1997
<b>decaf::lang::exceptions::IllegalStateException</b>	1998
<b>decaf::lang::exceptions::IllegalThreadStateException</b>	2001
<b>activemq::transport::inactivity::InactivityMonitor</b>	2004
<b>decaf::lang::exceptions::IndexOutOfBoundsException</b>	2008
<b>decaf::net::Inet4Address</b>	2011
<b>decaf::net::Inet6Address</b>	2015
<b>decaf::net::InetAddress</b> (Represents an IP address )	2016
<b>decaf::net::InetSocketAddress</b>	2023
<b>inflate_state</b>	2024
<b>decaf::util::zip::Inflater</b> (This class uncompresses data that was compressed using the <i>DEFLATE</i> algorithm (see <i>specification</i> ) )	2027
<b>decaf::util::zip::InflaterInputStream</b> (A <i>FilterInputStream</i> that decompresses data read from the wrapped <i>InputStream</i> instance )	2035
<b>decaf::io::InputStream</b> (A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes )	2043
<b>decaf::io::InputStreamReader</b> (An <i>InputStreamReader</i> (p. 2053) is a bridge from byte streams to character streams )	2053
<b>decaf::internal::nio::IntArrayBuffer</b>	2056
<b>decaf::nio::IntBuffer</b> (This class defines four categories of operations upon int buffers: )	2066
<b>decaf::lang::Integer</b>	2077
<b>activemq::commands::IntegerResponse</b>	2091
<b>activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2094) )	2094
<b>activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2098) )	2098
<b>activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2102) )	2102
<b>activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2106) )	2106
<b>activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2110) )	2110
<b>activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller</b> (Marshaling <i>code</i> (p. 1183) for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2114) )	2114
<b>internal_state</b>	2118
<b>activemq::transport::mock::InternalCommandListener</b> (Listens for Commands sent from the <i>MockTransport</i> (p. 2768) )	2122
<b>decaf::lang::exceptions::InterruptedException</b>	2124
<b>decaf::io::InterruptedException</b>	2127
<b>cms::InvalidClientIdException</b> (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider )	2130
<b>cms::InvalidDestinationException</b> (This exception must be thrown when a destination either is not understood by a provider or is no longer valid )	2131
<b>decaf::security::InvalidKeyException</b>	2132
<b>decaf::nio::InvalidMarkException</b>	2135



<b>cms::InvalidSelectorException</b> (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax ) . . . . .	2138
<b>decaf::lang::exceptions::InvalidStateException</b> . . . . .	2139
<b>decaf::io::IOException</b> . . . . .	2142
<b>activemq::transport::IOTransport</b> (Implementation of the <b>Transport</b> (p. 3883) interface that performs marshaling of <b>commands</b> (p. 87) to IO streams ) . . . .	2145
<b>decaf::lang::Iterable&lt; E &gt;</b> (Implementing this interface allows an object to be cast to an <b>Iterable</b> (p. 2152) type for generic collections API calls ) . . . . .	2152
<b>decaf::util::Iterator&lt; T &gt;</b> (Defines an object that can be used to iterate over the elements of a collection ) . . . . .	2154
<b>activemq::commands::JournalQueueAck</b> . . . . .	2156
<b>activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2159) ) . . . . .	2159
<b>activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2163) ) . . . . .	2163
<b>activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2167) ) . . . . .	2167
<b>activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2171) ) . . . . .	2171
<b>activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2175) ) . . . . .	2175
<b>activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2179) ) . . . . .	2179
<b>activemq::commands::JournalTopicAck</b> . . . . .	2183
<b>activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2188) ) . . . . .	2188
<b>activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2192) ) . . . . .	2192
<b>activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2196) ) . . . . .	2196
<b>activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2200) ) . . . . .	2200
<b>activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2204) ) . . . . .	2204
<b>activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2208) ) . . . . .	2208
<b>activemq::commands::JournalTrace</b> . . . . .	2212
<b>activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2215) ) . . . . .	2215

<b>activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2219) ) . . . . .	2219
<b>activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2223) ) . . . . .	2223
<b>activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2227) ) . . . . .	2227
<b>activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2231) ) . . . . .	2231
<b>activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2235) ) . . . . .	2235
<b>activemq::commands::JournalTransaction</b> . . . . .	2239
<b>activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2242) ) . . . . .	2242
<b>activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2246) ) . . . . .	2246
<b>activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2250) ) . . . . .	2250
<b>activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2254) ) . . . . .	2254
<b>activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2258) ) . . . . .	2258
<b>activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2262) ) . . . . .	2262
<b>activemq::commands::KeepAliveInfo</b> . . . . .	2266
<b>activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2269) ) . . . . .	2269
<b>activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2273) ) . . . . .	2273
<b>activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2277) ) . . . . .	2277
<b>activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2281) ) . . . . .	2281
<b>activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2285) ) . . . . .	2285
<b>activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2289) ) . . . . .	2289
<b>decaf::security::Key</b> (The <b>Key</b> (p.2293) interface is the top-level interface for all keys )	2293

<b>decaf::security::KeyException</b> . . . . .	2295
<b>decaf::security::KeyManagementException</b> . . . . .	2298
<b>activemq::commands::LastPartialCommand</b> . . . . .	2301
<b>activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2304) ) . . . . .	2304
<b>activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2308) ) . . . . .	2308
<b>activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2312) ) . . . . .	2312
<b>activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2316) ) . . . . .	2316
<b>activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2320) ) . . . . .	2320
<b>activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2324) ) . . . . .	2324
<b>decaf::util::comparators::Less&lt; E &gt;</b> (Simple <b>Less</b> (p. 2328) <b>Comparator</b> (p. 1217) that compares to elements to determine if the first is less than the second ) . .	2328
<b>std::less&lt; decaf::lang::ArrayPointer&lt; T &gt; &gt;</b> (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2330
<b>std::less&lt; decaf::lang::Pointer&lt; T &gt; &gt;</b> (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2331
<b>decaf::util::logging::Level</b> (Defines a set of standard <b>logging</b> (p. 175) levels that can be used to control <b>logging</b> (p. 175) output ) . . . . .	2332
<b>decaf::util::List&lt; E &gt;</b> (An ordered collection (also known as a sequence) ) . . . . .	2337
<b>decaf::util::ListIterator&lt; E &gt;</b> (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list ) . . . . .	2344
<b>activemq::commands::LocalTransactionId</b> . . . . .	2347
<b>activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2351) ) . . . . .	2351
<b>activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2355) ) . . . . .	2355
<b>activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2359) ) . . . . .	2359
<b>activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2363) ) . . . . .	2363
<b>activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2367) ) . . . . .	2367
<b>activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2371) ) . . . . .	2371
<b>decaf::util::concurrent::Lock</b> (A wrapper class around a given synchronization mech- anism that provides automatic release upon destruction ) . . . . .	2375

<b>decaf::util::concurrent::locks::Lock</b> ( <b>Lock</b> (p. 2377) implementations provide more extensive locking operations than can be obtained using synchronized statements )	2377
<b>decaf::util::concurrent::locks::LockSupport</b> (Basic thread blocking primitives for creating <b>locks</b> (p. 174) and other synchronization classes )	2383
<b>decaf::util::logging::Logger</b> (A <b>Logger</b> (p. 2386) object is used to log messages for a specific system or application component )	2386
<b>decaf::util::logging::LoggerHierarchy</b>	2398
<b>activemq::io::LoggingInputStream</b>	2399
<b>activemq::io::LoggingOutputStream</b> (OutputStream filter that just logs the data being written )	2401
<b>activemq::transport::logging::LoggingTransport</b> (A <b>transport</b> (p. 97) filter that logs <b>commands</b> (p. 87) as they are sent/received )	2403
<b>decaf::util::logging::LogManager</b> (There is a single global <b>LogManager</b> (p. 2406) object that is used to maintain a set of shared state about Loggers and log services )	2406
<b>decaf::util::logging::LogRecord</b> ( <b>LogRecord</b> (p. 2413) objects are used to pass <b>logging</b> (p. 175) requests between the <b>logging</b> (p. 175) framework and individual log Handlers )	2413
<b>decaf::util::logging::LogWriter</b>	2418
<b>decaf::lang::Long</b>	2420
<b>decaf::internal::nio::LongArrayBuffer</b>	2434
<b>decaf::nio::LongBuffer</b> (This class defines four categories of operations upon long long buffers: )	2444
<b>activemq::util::LongSequenceGenerator</b> (This class is used to generate a sequence of long long values that are incremented each time a new value is requested )	2455
<b>decaf::net::MalformedURLException</b>	2456
<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2459) template that wraps around a <b>std::map</b> to provide a more user-friendly interface and to provide common functions that do not exist in <b>std::map</b> )	2459
<b>cms::MapMessage</b> (A <b>MapMessage</b> (p. 2472) object is used to send a set of name-value pairs )	2472
<b>decaf::util::logging::MarkBlockLogger</b> (Defines a class that can be used to mark the entry and exit from scoped blocks )	2483
<b>activemq::wireformat::MarshalAware</b>	2484
<b>activemq::wireformat::openwire::marshal::v6::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2487
<b>activemq::wireformat::openwire::marshal::v3::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2488
<b>activemq::wireformat::openwire::marshal::v4::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2489
<b>activemq::wireformat::openwire::marshal::v5::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2490
<b>activemq::wireformat::openwire::marshal::v1::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2491
<b>activemq::wireformat::openwire::marshal::v2::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol )	2492
<b>activemq::util::MarshallingSupport</b>	2493
<b>decaf::lang::Math</b> (The class <b>Math</b> (p. 2497) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions )	2497
<b>activemq::util::MemoryUsage</b>	2512
<b>activemq::commands::Message</b>	2516
<b>cms::Message</b> (Root of all messages )	2534
<b>activemq::commands::MessageAck</b>	2559

<b>activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2565) ) . . . . .	2565
<b>activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2569) ) . . . . .	2569
<b>activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2573) ) . . . . .	2573
<b>activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2577) ) . . . . .	2577
<b>activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2581) ) . . . . .	2581
<b>activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2585) ) . . . . .	2585
<b>cms::MessageConsumer</b> (A client uses a <b>MessageConsumer</b> (p. 2589) to received messages from a destination ) . . . . .	2589
<b>activemq::cmsutil::MessageCreator</b> (Creates the user-defined message to be sent by the <b>CmsTemplate</b> (p. 1170) ) . . . . .	2593
<b>activemq::commands::MessageDispatch</b> . . . . .	2594
<b>activemq::core::MessageDispatchChannel</b> . . . . .	2599
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2606) ) . . . . .	2606
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2610) ) . . . . .	2610
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2614) ) . . . . .	2614
<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2618) ) . . . . .	2618
<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2622) ) . . . . .	2622
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2626) ) . . . . .	2626
<b>activemq::commands::MessageDispatchNotification</b> . . . . .	2630
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2635) ) . . . . .	2635
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2639) ) . . . . .	2639
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2643) ) . . . . .	2643

<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2647) ) . . . . .	2647
<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2651) ) . . . . .	2651
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2655) ) . . . . .	2655
<b>cms::MessageEnumeration</b> (Defines an object that enumerates a collection of Messages ) . . . . .	2659
<b>cms::MessageEOFException</b> (This exception must be thrown when an unexpected end of stream has been reached when a <b>StreamMessage</b> (p. 3652) or <b>BytesMessage</b> (p. 1056) is being read ) . . . . .	2661
<b>cms::MessageFormatException</b> (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type ) . . . . .	2662
<b>activemq::commands::MessageId</b> . . . . .	2663
<b>activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2668) ) . . . . .	2668
<b>activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2672) ) . . . . .	2672
<b>activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2676) ) . . . . .	2676
<b>activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2680) ) . . . . .	2680
<b>activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2684) ) . . . . .	2684
<b>activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2688) ) . . . . .	2688
<b>cms::MessageListener</b> (A <b>MessageListener</b> (p. 2692) object is used to receive asynchronously delivered messages ) . . . . .	2692
<b>activemq::wireformat::openwire::marshal::v5::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2693) ) . . . . .	2693
<b>activemq::wireformat::openwire::marshal::v3::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2698) ) . . . . .	2698
<b>activemq::wireformat::openwire::marshal::v2::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2703) ) . . . . .	2703
<b>activemq::wireformat::openwire::marshal::v4::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2708) ) . . . . .	2708
<b>activemq::wireformat::openwire::marshal::v1::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2713) ) . . . . .	2713
<b>activemq::wireformat::openwire::marshal::v6::MessageMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>MessageMarshaller</b> (p. 2718) ) . . . . .	2718
<b>cms::MessageNotReadableException</b> (This exception must be thrown when a CMS client attempts to read a write-only message ) . . . . .	2723
<b>cms::MessageNotWritableException</b> (This exception must be thrown when a CMS client attempts to write to a read-only message ) . . . . .	2724

<b>cms::MessageProducer</b> (A client uses a <b>MessageProducer</b> (p. 2725) object to send messages to a <b>Destination</b> (p. 1723) ) . . . . .	2725
<b>activemq::wireformat::openwire::utils::MessagePropertyInterceptor</b> (Used the base <b>ActiveMQMessage</b> class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the <b>OpenWire</b> Message properties ) . . . . .	2732
<b>activemq::commands::MessagePull</b> . . . . .	2739
<b>activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2744) ) . . . . .	2744
<b>activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2748) ) . . . . .	2748
<b>activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2752) ) . . . . .	2752
<b>activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2756) ) . . . . .	2756
<b>activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2760) ) . . . . .	2760
<b>activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>MessagePullMarshaller</b> (p. 2764) ) . . . . .	2764
<b>activemq::transport::mock::MockTransport</b> (The <b>MockTransport</b> (p. 2768) defines a base level <b>Transport</b> (p. 3883) class that is intended to be used in place of an a regular protocol <b>Transport</b> (p. 3883) such as TCP ) . . . . .	2768
<b>activemq::transport::mock::MockTransportFactory</b> (Manufactures <b>MockTransports</b> , which are objects that read from input streams and write to output streams ) . . . . .	2779
<b>decaf::util::concurrent::Mutex</b> ( <b>Mutex</b> (p. 2782) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java ) . . . . .	2782
<b>decaf::util::concurrent::MutexHandle</b> . . . . .	2787
<b>decaf::internal::util::concurrent::MutexImpl</b> . . . . .	2788
<b>decaf::internal::net::Network</b> (Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API ) . . . . .	2790
<b>activemq::commands::NetworkBridgeFilter</b> . . . . .	2793
<b>activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>NetworkBridgeFilterMarshaller</b> (p. 2796) ) . . . . .	2796
<b>activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>NetworkBridgeFilterMarshaller</b> (p. 2800) ) . . . . .	2800
<b>activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>NetworkBridgeFilterMarshaller</b> (p. 2804) ) . . . . .	2804
<b>activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>NetworkBridgeFilterMarshaller</b> (p. 2808) ) . . . . .	2808
<b>activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for <b>Open Wire Format</b> for <b>NetworkBridgeFilterMarshaller</b> (p. 2812) ) . . . . .	2812

<b>activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2816) ) . . . . .	2816
<b>decaf::net::NoRouteToHostException</b> . . . . .	2820
<b>decaf::security::NoSuchAlgorithmException</b> . . . . .	2823
<b>decaf::lang::exceptions::NoSuchElementException</b> . . . . .	2826
<b>decaf::security::NoSuchProviderException</b> . . . . .	2829
<b>decaf::lang::exceptions::NullPointerException</b> . . . . .	2832
<b>decaf::lang::Number</b> (The abstract class <b>Number</b> (p. 2835) is the superclass of classes <b>Byte</b> (p. 960), <b>Double</b> (p. 1788), <b>Float</b> (p. 1904), <b>Integer</b> (p. 2077), <b>Long</b> (p. 2420), and <b>Short</b> (p. 3440) ) . . . . .	2835
<b>decaf::lang::exceptions::NumberFormatException</b> . . . . .	2838
<b>cms::ObjectMessage</b> (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object ) . . .	2841
<b>decaf::internal::net::ssl::openssl::OpenSSLContextSpi</b> (Provides an <b>SSLContext</b> that wraps the <b>OpenSSL</b> API ) . . . . .	2842
<b>decaf::internal::net::ssl::openssl::OpenSSLParameters</b> (Container class for parameters that are Common to <b>OpenSSL</b> socket classes ) . . . . .	2845
<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocket</b> ( <b>SSLServerSocket</b> based on <b>OpenSSL</b> library <b>code</b> (p. 1183) ) . . . . .	2847
<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory</b> ( <b>SSLServerSocketFactory</b> that creates <b>Server Sockets</b> that use <b>OpenSSL</b> ) . . . . .	2853
<b>decaf::internal::net::ssl::openssl::OpenSSLSocket</b> (Wraps a a Normal <b>Socket</b> object and extends or overrides functions in that class to make use of the <b>OpenSSL</b> <b>Socket</b> API ) . . . . .	2858
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketException</b> (Subclass of the standard <b>SocketException</b> that knows how to produce an error message from the <b>OpenSSL</b> error stack ) . . . . .	2871
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketFactory</b> (Client <b>Socket</b> <b>Factory</b> that creates <b>SSL</b> based client sockets using the <b>OpenSSL</b> library ) . . . . .	2875
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream</b> (An output stream for reading data from an <b>OpenSSL</b> <b>Socket</b> instance ) . . . . .	2882
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream</b> ( <b>OutputStream</b> implementation used to write data to an <b>OpenSSLSocket</b> (p. 2858) instance )	2885
<b>activemq::wireformat::openwire::OpenWireFormat</b> . . . . .	2887
<b>activemq::wireformat::openwire::OpenWireFormatFactory</b> . . . . .	2899
<b>activemq::wireformat::openwire::OpenWireFormatNegotiator</b> . . . . .	2901
<b>activemq::wireformat::openwire::OpenWireResponseBuilder</b> . . . . .	2905
<b>decaf::io::OutputStream</b> (Base interface for any class that wants to represent an output stream of bytes ) . . . . .	2907
<b>decaf::io::OutputStreamWriter</b> (A class for turning a character stream into a byte stream ) . . . . .	2915
<b>activemq::commands::PartialCommand</b> . . . . .	2918
<b>activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2922) ) . . . . .	2922
<b>activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2926) ) . . . . .	2926
<b>activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2930) ) . . . . .	2930



<b>activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p.2934) ) . . . . .	2934
<b>activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p.2938) ) . . . . .	2938
<b>activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>PartialCommandMarshaller</b> (p.2942) ) . . . . .	2942
<b>decaf::lang::Pointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p.2946) that is a template on a Type and is <b>Thread</b> (p.3765) Safe if the default Reference Counter is used ) . . . . .	2946
<b>decaf::lang::PointerComparator&lt; T, R &gt;</b> (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the <b>Pointer</b> (p.2946) instance ) . . . . .	2954
<b>activemq::cmsutil::PooledSession</b> (A pooled session object that wraps around a delegate session ) . . . . .	2955
<b>decaf::util::concurrent::PooledThread</b> . . . . .	2968
<b>decaf::util::concurrent::PooledThreadListener</b> (Abstract Listener Interface for users of <b>ThreadPool</b> (p.3777) ) . . . . .	2971
<b>decaf::net::PortUnreachableException</b> . . . . .	2973
<b>activemq::core::PrefetchPolicy</b> (Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP ) . . . . .	2976
<b>activemq::util::PrimitiveList</b> (List of primitives ) . . . . .	2980
<b>activemq::util::PrimitiveMap</b> (Map of named primitives ) . . . . .	2991
<b>activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller</b> (This class wraps the functionality needed to <b>marshal</b> (p.107) a primitive map to the Openwire Format's expectation of what the map looks like on the wire ) . .	3001
<b>activemq::util::PrimitiveValueNode::PrimitiveValue</b> (Define a union type comprised of the various types ) . . . . .	3008
<b>activemq::util::PrimitiveValueConverter</b> (Class controls the conversion of data contained in a <b>PrimitiveValueNode</b> (p.3012) from one type to another ) . . . .	3010
<b>activemq::util::PrimitiveValueNode</b> (Class that wraps around a single value of one of the many types ) . . . . .	3012
<b>decaf::security::Principal</b> (Base interface for a principal, which can represent an individual or organization ) . . . . .	3026
<b>decaf::util::PriorityQueue&lt; E &gt;</b> (An unbounded priority queue based on a binary heap algorithm ) . . . . .	3028
<b>activemq::commands::ProducerAck</b> . . . . .	3036
<b>activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3040) ) . . . . .	3040
<b>activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3044) ) . . . . .	3044
<b>activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3048) ) . . . . .	3048
<b>activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3052) ) . . . . .	3052

<b>activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3056) )	3056
<b>activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerAckMarshaller</b> (p.3060) )	3060
<b>activemq::cmsutil::ProducerCallback</b> (Callback for sending a message to a CMS destination )	3064
<b>activemq::cmsutil::CmsTemplate::ProducerExecutor</b>	3065
<b>activemq::commands::ProducerId</b>	3067
<b>activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3072) )	3072
<b>activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3076) )	3076
<b>activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3080) )	3080
<b>activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3084) )	3084
<b>activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3088) )	3088
<b>activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerIdMarshaller</b> (p.3092) )	3092
<b>activemq::commands::ProducerInfo</b>	3096
<b>activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3101) )	3101
<b>activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3105) )	3105
<b>activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3109) )	3109
<b>activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3113) )	3113
<b>activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3117) )	3117
<b>activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ProducerInfoMarshaller</b> (p.3121) )	3121
<b>activemq::state::ProducerState</b>	3125
<b>decaf::util::Properties</b> (Java-like properties class for mapping string names to string values )	3126
<b>decaf::util::logging::PropertiesChangeListener</b> (Defines the interface that classes can use to listen for change events on <b>Properties</b> (p.3126) )	3135
<b>decaf::net::ProtocolException</b>	3137
<b>decaf::security::PublicKey</b> (A public key )	3140

<b>decaf::io::PushbackInputStream</b> (A <b>PushbackInputStream</b> (p. 3141) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte ) . . . . .	3141
<b>cms::Queue</b> (An interface encapsulating a provider-specific queue name ) . . . . .	3148
<b>decaf::util::Queue&lt; E &gt;</b> (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection ) . . . . .	3149
<b>cms::QueueBrowser</b> (This class implements in interface for browsing the messages in a <b>Queue</b> (p. 3148) without removing them ) . . . . .	3153
<b>decaf::util::Random</b> ( <b>Random</b> (p. 3155) Value Generator which is used to generate a stream of pseudorandom numbers ) . . . . .	3155
<b>decaf::lang::Readable</b> (A <b>Readable</b> (p. 3160) is a source of characters ) . . . . .	3160
<b>activemq::transport::inactivity::ReadChecker</b> (Runnable class that is used by the { ) . . . . .	3162
<b>decaf::io::Reader</b> . . . . .	3163
<b>decaf::nio::ReadOnlyBufferException</b> . . . . .	3169
<b>decaf::util::concurrent::locks::ReadWriteLock</b> (A <b>ReadWriteLock</b> (p. 3172) maintains a pair of associated <b>locks</b> (p. 174), one for read-only operations and one for writing ) . . . . .	3172
<b>activemq::cmsutil::CmsTemplate::ReceiveExecutor</b> . . . . .	3174
<b>activemq::core::RedeliveryPolicy</b> (Interface for a <b>RedeliveryPolicy</b> (p. 3177) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back ) . . . . .	3177
<b>decaf::util::concurrent::locks::ReentrantLock</b> (A reentrant mutual exclusion <b>Lock</b> (p. 2377) with extended capabilities ) . . . . .	3182
<b>decaf::util::concurrent::RejectedExecutionException</b> . . . . .	3189
<b>decaf::util::concurrent::RejectedExecutionHandler</b> (A handler for tasks that cannot be executed by a <b>ThreadPoolExecutor</b> (p. ??) ) . . . . .	3192
<b>activemq::commands::RemoveInfo</b> . . . . .	3194
<b>activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3198) ) . . . . .	3198
<b>activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3202) ) . . . . .	3202
<b>activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3206) ) . . . . .	3206
<b>activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3210) ) . . . . .	3210
<b>activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3214) ) . . . . .	3214
<b>activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3218) ) . . . . .	3218
<b>activemq::commands::RemoveSubscriptionInfo</b> . . . . .	3222
<b>activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3227) ) . . . . .	3227
<b>activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3231) ) . . . . .	3231

<b>activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3235) ) . . . . .	3235
<b>activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3239) ) . . . . .	3239
<b>activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3243) ) . . . . .	3243
<b>activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3247) ) . . . . .	3247
<b>activemq::commands::ReplayCommand</b> . . . . .	3251
<b>activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3254) ) . . . . .	3254
<b>activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3258) ) . . . . .	3258
<b>activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3262) ) . . . . .	3262
<b>activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3266) ) . . . . .	3266
<b>activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3270) ) . . . . .	3270
<b>activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ReplayCommand-</b> <b>Marshaller</b> (p. 3274) ) . . . . .	3274
<b>activemq::cmsutil::CmsTemplate::ResolveProducerExecutor</b> . . . . .	3278
<b>activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor</b> . . . . .	3279
<b>decaf::internal::util::Resource</b> (Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown) . .	3280
<b>decaf::internal::util::ResourceLifecycleManager</b> . . . . .	3281
<b>activemq::cmsutil::ResourceLifecycleManager</b> (Manages the lifecycle of a set of CMS resources) . . . . .	3282
<b>activemq::commands::Response</b> . . . . .	3285
<b>activemq::transport::mock::ResponseBuilder</b> (Interface for all Protocols to imple- ment that defines the behavior of the Broker in response to messages of that protocol) . . . . .	3289
<b>activemq::transport::correlator::ResponseCorrelator</b> (This type of <b>transport</b> (p. 97) filter is responsible for correlating asynchronous responses with requests) .	3291
<b>activemq::wireformat::openwire::marshal::v4::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3295) ) . .	3295
<b>activemq::wireformat::openwire::marshal::v2::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3300) ) . .	3300
<b>activemq::wireformat::openwire::marshal::v5::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3305) ) . .	3305
<b>activemq::wireformat::openwire::marshal::v3::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3310) ) . .	3310
<b>activemq::wireformat::openwire::marshal::v1::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3315) ) . .	3315

<b>activemq::wireformat::openwire::marshal::v6::ResponseMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>ResponseMarshaller</b> (p. 3320) ) . .	3320
<b>decaf::lang::Runnable</b> (Interface for a runnable object - defines a task that can be run by a thread ) . . . . .	3325
<b>decaf::lang::Runtime</b> . . . . .	3326
<b>decaf::lang::exceptions::RuntimeException</b> . . . . .	3328
<b>decaf::security::SecureRandom</b> . . . . .	3331
<b>decaf::internal::security::SecureRandomImpl</b> (Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources ) . . . . .	3336
<b>decaf::security::SecureRandomSpi</b> (Interface class used by Security Service Providers to implement a source of secure random bytes ) . . . . .	3339
<b>decaf::util::concurrent::Semaphore</b> (A counting semaphore ) . . . . .	3341
<b>activemq::cmsutil::CmsTemplate::SendExecutor</b> . . . . .	3350
<b>decaf::net::ServerSocket</b> (This class implements server sockets ) . . . . .	3352
<b>decaf::net::ServerSocketFactory</b> (Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies )	3361
<b>cms::Session</b> (A <b>Session</b> (p. 3365) object is a single-threaded context for producing and consuming messages ) . . . . .	3365
<b>activemq::cmsutil::SessionCallback</b> (Callback for executing any number of operations on a provided CMS Session ) . . . . .	3378
<b>activemq::commands::SessionId</b> . . . . .	3379
<b>activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3383) ) . .	3383
<b>activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3387) ) . .	3387
<b>activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3391) ) . .	3391
<b>activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3395) ) . .	3395
<b>activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3399) ) . .	3399
<b>activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3403) ) . .	3403
<b>activemq::commands::SessionInfo</b> . . . . .	3407
<b>activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3411) ) . . . . .	3411
<b>activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3415) ) . . . . .	3415
<b>activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3419) ) . . . . .	3419
<b>activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3423) ) . . . . .	3423
<b>activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3427) ) . . . . .	3427
<b>activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller</b> (Marshaling code (p. 1183) for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3431) ) . . . . .	3431

<b>activemq::cmsutil::SessionPool</b> (A pool of CMS sessions from the same connection and with the same acknowledge mode )	3435
<b>activemq::state::SessionState</b>	3437
<b>decaf::util::Set&lt; E &gt;</b> (A collection that contains no duplicate elements )	3439
<b>decaf::lang::Short</b>	3440
<b>decaf::internal::nio::ShortArrayBuffer</b>	3449
<b>decaf::nio::ShortBuffer</b> (This class defines four categories of operations upon short buffers: )	3459
<b>activemq::commands::ShutdownInfo</b>	3470
<b>activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3473) )	3473
<b>activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3477) )	3477
<b>activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3481) )	3481
<b>activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3485) )	3485
<b>activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3489) )	3489
<b>activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller</b> (Marshaling <b>code</b> (p.1183) for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p.3493) )	3493
<b>decaf::security::SignatureException</b>	3497
<b>decaf::util::logging::SimpleFormatter</b> (Print a brief summary of the <b>LogRecord</b> (p.2413) in a human readable format )	3500
<b>decaf::util::logging::SimpleLogger</b>	3501
<b>decaf::net::Socket</b>	3503
<b>decaf::net::SocketAddress</b> (Base class for protocol specific <b>Socket</b> (p.3503) addresses )	3519
<b>decaf::net::SocketError</b> (Static utility class to simplify handling of error codes for socket operations )	3520
<b>decaf::net::SocketException</b> (Exception for errors when manipulating sockets )	3521
<b>decaf::net::SocketFactory</b> (The <b>SocketFactory</b> (p.3524) is used to create <b>Socket</b> (p.3503) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations )	3524
<b>decaf::internal::net::SocketFileDescriptor</b> (File Descriptor type used internally by Decaf Socket objects )	3528
<b>decaf::net::SocketImpl</b> (Acts as a base class for all physical <b>Socket</b> (p.3503) implementations )	3529
<b>decaf::net::SocketImplFactory</b> (Factory class interface for a Factory that creates SocketImpl objects )	3537
<b>decaf::net::SocketOptions</b>	3538
<b>decaf::net::SocketTimeoutException</b>	3542
<b>decaf::net::ssl::SSLContext</b> (Represents on implementation of the Secure <b>Socket</b> (p.3503) Layer for streaming based sockets )	3545
<b>decaf::net::ssl::SSLContextSpi</b> (Defines the interface that should be provided by an <b>SSLContext</b> (p.3545) provider )	3548
<b>decaf::net::ssl::SSLParameters</b>	3551

<b>decaf::net::ssl::SSLServerSocket</b> (Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol ) . . . . .	3554
<b>decaf::net::ssl::SSLServerSocketFactory</b> (Factory class interface that provides methods to create SSL Server Sockets ) . . . . .	3560
<b>decaf::net::ssl::SSLSocket</b> . . . . .	3563
<b>decaf::net::ssl::SSLSocketFactory</b> (Factory class interface for a <b>SocketFactory</b> (p. 3524) that can create <b>SSLSocket</b> (p. 3563) objects ) . . . . .	3571
<b>activemq::transport::tcp::SslTransport</b> ( <b>Transport</b> (p. 3883) for connecting to a Broker using an SSL Socket ) . . . . .	3574
<b>activemq::transport::tcp::SslTransportFactory</b> . . . . .	3576
<b>activemq::commands::BrokerError::StackTraceElement</b> . . . . .	3578
<b>decaf::internal::io::StandardErrorOutputStream</b> (Wrapper Around the Standard error Output facility on the current platform ) . . . . .	3579
<b>decaf::internal::io::StandardInputStream</b> . . . . .	3582
<b>decaf::internal::io::StandardOutputStream</b> . . . . .	3584
<b>cms::Startable</b> (Interface for a class that implements the start method ) . . . . .	3586
<b>decaf::lang::STATIC_CAST_TOKEN</b> . . . . .	3587
<b>activemq::core::ActiveMQConstants::StaticInitializer</b> . . . . .	3588
<b>decaf::util::StlList&lt; E &gt;</b> ( <b>List</b> (p. 2337) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type ) . . . . .	3589
<b>decaf::util::StlMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map ) . . . . .	3602
<b>decaf::util::StlQueue&lt; T &gt;</b> (The <b>Queue</b> (p. 3149) class accepts messages with an psuh(m) command where m is the message to be queued ) . . . . .	3615
<b>decaf::util::StlSet&lt; E &gt;</b> ( <b>Set</b> (p. 3439) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set ) . . . . .	3623
<b>activemq::wireformat::stomp::StompCommandConstants</b> . . . . .	3629
<b>activemq::wireformat::stomp::StompFrame</b> (A Stomp-level message frame that encloses all messages to and from the broker ) . . . . .	3633
<b>activemq::wireformat::stomp::StompHelper</b> (Utility Methods used when marshaling to and from StompFrame's ) . . . . .	3638
<b>activemq::wireformat::stomp::StompWireFormat</b> . . . . .	3643
<b>activemq::wireformat::stomp::StompWireFormatFactory</b> (Factory used to create the Stomp Wire Format instance ) . . . . .	3647
<b>cms::Stoppable</b> (Interface for a class that implements the stop method ) . . . . .	3648
<b>decaf::util::logging::StreamHandler</b> (Stream based <b>logging</b> (p. 175) <b>Handler</b> (p. 1978) ) . . . . .	3649
<b>cms::StreamMessage</b> (Interface for a <b>StreamMessage</b> (p. 3652) ) . . . . .	3652
<b>decaf::lang::String</b> (Immutable sequence of chars ) . . . . .	3665
<b>decaf::util::StringTokenizer</b> . . . . .	3668
<b>activemq::commands::SubscriptionInfo</b> . . . . .	3671
<b>activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3676) ) . . . . .	3676
<b>activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3680) ) . . . . .	3680
<b>activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3684) ) . . . . .	3684

<b>activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3688) ) . . . . .	3688
<b>activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3692) ) . . . . .	3692
<b>activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3696) ) . . . . .	3696
<b>decaf::util::concurrent::Synchronizable</b> (The interface for all synchronizable objects (that is, objects that can be locked and unlocked) ) . . . . .	3700
<b>decaf::internal::util::concurrent::SynchronizableImpl</b> (A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance) . . . . .	3711
<b>activemq::core::Synchronization</b> (Transacted Object <b>Synchronization</b> (p. 3715), used to sync the events of a Transaction with the items in the Transaction) . . . . .	3715
<b>decaf::util::concurrent::SynchronousQueue&lt; E &gt;</b> (A <b>blocking queue</b> (p. 843) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa) . . . . .	3716
<b>decaf::lang::System</b> (Static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays) . . . . .	3726
<b>activemq::threads::Task</b> (Represents a unit of work that requires one or more iterations to complete) . . . . .	3734
<b>decaf::util::concurrent::TaskListener</b> . . . . .	3735
<b>activemq::threads::TaskRunner</b> . . . . .	3736
<b>decaf::internal::net::tcp::TcpSocket</b> (Platform-independent implementation of the socket interface) . . . . .	3738
<b>decaf::internal::net::tcp::TcpSocketInputStream</b> (Input stream for performing reads on a socket) . . . . .	3747
<b>decaf::internal::net::tcp::TcpSocketOutputStream</b> (Output stream for performing write operations on a socket) . . . . .	3750
<b>activemq::transport::tcp::TcpTransport</b> (Implements a TCP/IP based <b>transport</b> (p. 97) filter, this <b>transport</b> (p. 97) is meant to wrap an instance of an <b>IOTransport</b> (p. 2145) ) . . . . .	3752
<b>activemq::transport::tcp::TcpTransportFactory</b> (Factory Responsible for creating the <b>TcpTransport</b> (p. 3752) ) . . . . .	3756
<b>cms::TemporaryQueue</b> (Defines a Temporary <b>Queue</b> (p. 3148) based <b>Destination</b> (p. 1723) ) . . . . .	3759
<b>cms::TemporaryTopic</b> (Defines a Temporary <b>Topic</b> (p. 3817) based <b>Destination</b> (p. 1723) ) . . . . .	3761
<b>cms::TextMessage</b> (Interface for a text message) . . . . .	3763
<b>decaf::lang::Thread</b> (A <b>Thread</b> (p. 3765) is a concurrent unit of execution) . . . . .	3765
<b>decaf::util::concurrent::ThreadFactory</b> (Public interface <b>ThreadFactory</b> (p. 3774) ) . . . . .	3774
<b>decaf::lang::ThreadGroup</b> . . . . .	3776
<b>decaf::util::concurrent::ThreadPool</b> (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks) . . . . .	3777
<b>decaf::lang::Throwable</b> (This class represents an error that has occurred) . . . . .	3783
<b>decaf::util::concurrent::TimeoutException</b> . . . . .	3787
<b>decaf::util::Timer</b> (A facility for threads to schedule tasks for future execution in a background thread) . . . . .	3790
<b>decaf::util::TimerTask</b> (A Base class for a task object that can be scheduled for one-time or repeated execution by a <b>Timer</b> (p. 3790) ) . . . . .	3801



<b>decaf::internal::util::TimerTaskHeap</b> (A Binary Heap implemented specifically for the Timer class in Decaf Util ) . . . . .	3804
<b>decaf::util::concurrent::TimeUnit</b> (A <b>TimeUnit</b> (p. 3807) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units ) . . . . .	3807
<b>cms::Topic</b> (An interface encapsulating a provider-specific topic name ) . . . . .	3817
<b>activemq::state::Tracked</b> . . . . .	3818
<b>activemq::commands::TransactionId</b> . . . . .	3819
<b>activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3823) ) . . . . .	3823
<b>activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3827) ) . . . . .	3827
<b>activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3831) ) . . . . .	3831
<b>activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3835) ) . . . . .	3835
<b>activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3839) ) . . . . .	3839
<b>activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3843) ) . . . . .	3843
<b>activemq::commands::TransactionInfo</b> . . . . .	3847
<b>activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3851) ) . . . . .	3851
<b>activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3855) ) . . . . .	3855
<b>activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3859) ) . . . . .	3859
<b>activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3863) ) . . . . .	3863
<b>activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3867) ) . . . . .	3867
<b>activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3871) ) . . . . .	3871
<b>activemq::state::TransactionState</b> . . . . .	3875
<b>decaf::internal::util::concurrent::Transferer&lt; E &gt;</b> (Shared <b>internal</b> (p. 144) API for dual stacks and queues ) . . . . .	3877
<b>decaf::internal::util::concurrent::TransferQueue&lt; E &gt;</b> (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers ) . . . . .	3878
<b>decaf::internal::util::concurrent::TransferStack&lt; E &gt;</b> . . . . .	3881
<b>activemq::transport::Transport</b> (Interface for a <b>transport</b> (p. 97) layer for command objects ) . . . . .	3883

<b>activemq::transport::TransportFactory</b> (Defines the interface for Factories that create Transports or TransportFilters ) . . . . .	3889
<b>activemq::transport::TransportFilter</b> (A filter on the <b>transport</b> (p. 97) layer ) . . . . .	3891
<b>activemq::transport::TransportListener</b> (A listener of asynchronous <b>exceptions</b> (p. 92) from a command <b>transport</b> (p. 97) object ) . . . . .	3900
<b>activemq::transport::TransportRegistry</b> (Registry of all <b>Transport</b> (p. 3883) Factories that are available to the client at runtime ) . . . . .	3902
<b>tree_desc_s</b> . . . . .	3905
<b>decaf::lang::Thread::UncaughtExceptionHandler</b> (Interface for handlers invoked when a <b>Thread</b> (p. 3765) abruptly terminates due to an uncaught exception ) . . . . .	3906
<b>decaf::net::UnknownHostException</b> . . . . .	3907
<b>decaf::net::UnknownServiceException</b> . . . . .	3910
<b>decaf::io::UnsupportedEncodingException</b> (Thrown when the the Character Encoding is not supported ) . . . . .	3913
<b>decaf::lang::exceptions::UnsupportedOperationException</b> . . . . .	3916
<b>cms::UnsupportedOperationException</b> (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use ) . . . . .	3919
<b>decaf::net::URI</b> (This class represents an instance of a <b>URI</b> (p. 3921) as defined by RFC 2396 ) . . . . .	3921
<b>decaf::internal::net::URIEncoderDecoder</b> . . . . .	3932
<b>decaf::internal::net::URIHelper</b> (Helper class used by the URI classes in encoding and decoding of URI's ) . . . . .	3935
<b>activemq::transport::failover::URIPool</b> . . . . .	3942
<b>activemq::util::URISupport</b> . . . . .	3945
<b>decaf::net::URISyntaxException</b> . . . . .	3948
<b>decaf::internal::net::URIType</b> (Basic type object that holds data that composes a given URI ) . . . . .	3952
<b>decaf::net::URL</b> (Class <b>URL</b> (p. 3960) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web ) . . . . .	3960
<b>decaf::net::URLDecoder</b> . . . . .	3962
<b>decaf::net::URLEncoder</b> . . . . .	3963
<b>activemq::util::Usage</b> . . . . .	3964
<b>decaf::io::UTFDataFormatException</b> (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered ) . . . . .	3966
<b>decaf::util::UUID</b> (A class that represents an immutable universally unique identifier ( <b>UUID</b> (p. 3969)) ) . . . . .	3969
<b>activemq::wireformat::WireFormat</b> (Provides a mechanism to marshal <b>commands</b> (p. 87) into and out of packets or into and out of streams, Channels and Data-grams ) . . . . .	3976
<b>activemq::wireformat::WireFormatFactory</b> (The <b>WireFormatFactory</b> (p. 3980) is the interface that all <b>WireFormatFactory</b> (p. 3980) classes must extend ) . . . . .	3980
<b>activemq::commands::WireFormatInfo</b> . . . . .	3982
<b>activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3992) ) . . . . .	3992
<b>activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3996) ) . . . . .	3996
<b>activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 4000) ) . . . . .	4000

<b>activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 4004) ) . . . . .	4004
<b>activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 4008) ) . . . . .	4008
<b>activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 4012) ) . . . . .	4012
<b>activemq::wireformat::WireFormatNegotiator</b> (Defines a <b>WireFormatNegotiator</b> (p. 4016) which allows a <b>WireFormat</b> (p. 3976) to ) . . . . .	4016
<b>activemq::wireformat::WireFormatRegistry</b> (Registry of all <b>WireFormat</b> (p. 3976) Factories that are available to the client at runtime ) . . . . .	4017
<b>activemq::transport::inactivity::WriteChecker</b> (Runnable class used by the { ) . . . . .	4020
<b>decaf::io::Writer</b> . . . . .	4021
<b>decaf::security::auth::x500::X500Principal</b> . . . . .	4027
<b>decaf::security::cert::X509Certificate</b> (Base interface for all identity certificates ) . . . . .	4028
<b>activemq::commands::XATransactionId</b> . . . . .	4031
<b>activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4036) ) . . . . .	4036
<b>activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4040) ) . . . . .	4040
<b>activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4044) ) . . . . .	4044
<b>activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4048) ) . . . . .	4048
<b>activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4052) ) . . . . .	4052
<b>activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller</b> (Marshaling <b>code</b> (p. 1183) for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 4056) ) . . . . .	4056
<b>decaf::util::logging::XMLFormatter</b> (Format a <b>LogRecord</b> (p. 2413) into a standard XML format ) . . . . .	4060
<b>z_stream_s</b> . . . . .	4062
<b>decaf::util::zip::ZipException</b> . . . . .	4064



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/	<b>CachedConsumer.h</b>	4067
src/main/activemq/cmsutil/	<b>CachedProducer.h</b>	4068
src/main/activemq/cmsutil/	<b>CmsAccessor.h</b>	4069
src/main/activemq/cmsutil/	<b>CmsDestinationAccessor.h</b>	4070
src/main/activemq/cmsutil/	<b>CmsTemplate.h</b>	4071
src/main/activemq/cmsutil/	<b>DestinationResolver.h</b>	4072
src/main/activemq/cmsutil/	<b>DynamicDestinationResolver.h</b>	4073
src/main/activemq/cmsutil/	<b>MessageCreator.h</b>	4074
src/main/activemq/cmsutil/	<b>PooledSession.h</b>	4075
src/main/activemq/cmsutil/	<b>ProducerCallback.h</b>	4076
src/main/activemq/cmsutil/	<b>ResourceLifecycleManager.h</b>	4077
src/main/activemq/cmsutil/	<b>SessionCallback.h</b>	4079
src/main/activemq/cmsutil/	<b>SessionPool.h</b>	4080
src/main/activemq/commands/	<b>ActiveMQBlobMessage.h</b>	4081
src/main/activemq/commands/	<b>ActiveMQBytesMessage.h</b>	4082
src/main/activemq/commands/	<b>ActiveMQDestination.h</b>	4083
src/main/activemq/commands/	<b>ActiveMQMapMessage.h</b>	4084
src/main/activemq/commands/	<b>ActiveMQMessage.h</b>	4085
src/main/activemq/commands/	<b>ActiveMQMessageTemplate.h</b>	4086
src/main/activemq/commands/	<b>ActiveMQObjectMessage.h</b>	4087
src/main/activemq/commands/	<b>ActiveMQQueue.h</b>	4088
src/main/activemq/commands/	<b>ActiveMQStreamMessage.h</b>	4089
src/main/activemq/commands/	<b>ActiveMQTempDestination.h</b>	4090
src/main/activemq/commands/	<b>ActiveMQTempQueue.h</b>	4091
src/main/activemq/commands/	<b>ActiveMQTempTopic.h</b>	4092
src/main/activemq/commands/	<b>ActiveMQTextMessage.h</b>	4093
src/main/activemq/commands/	<b>ActiveMQTopic.h</b>	4094
src/main/activemq/commands/	<b>BaseCommand.h</b>	4095
src/main/activemq/commands/	<b>BaseDataStructure.h</b>	4096
src/main/activemq/commands/	<b>BooleanExpression.h</b>	4097
src/main/activemq/commands/	<b>BrokerError.h</b>	4098
src/main/activemq/commands/	<b>BrokerId.h</b>	4099
src/main/activemq/commands/	<b>BrokerInfo.h</b>	4100

src/main/activemq/commands/	<b>Command.h</b>	4101
src/main/activemq/commands/	<b>ConnectionControl.h</b>	4102
src/main/activemq/commands/	<b>ConnectionError.h</b>	4103
src/main/activemq/commands/	<b>ConnectionId.h</b>	4104
src/main/activemq/commands/	<b>ConnectionInfo.h</b>	4105
src/main/activemq/commands/	<b>ConsumerControl.h</b>	4106
src/main/activemq/commands/	<b>ConsumerId.h</b>	4107
src/main/activemq/commands/	<b>ConsumerInfo.h</b>	4108
src/main/activemq/commands/	<b>ControlCommand.h</b>	4109
src/main/activemq/commands/	<b>DataArrayResponse.h</b>	4110
src/main/activemq/commands/	<b>DataResponse.h</b>	4111
src/main/activemq/commands/	<b>DataStructure.h</b>	4112
src/main/activemq/commands/	<b>DestinationInfo.h</b>	4113
src/main/activemq/commands/	<b>DiscoveryEvent.h</b>	4114
src/main/activemq/commands/	<b>ExceptionResponse.h</b>	4115
src/main/activemq/commands/	<b>FlushCommand.h</b>	4116
src/main/activemq/commands/	<b>IntegerResponse.h</b>	4117
src/main/activemq/commands/	<b>JournalQueueAck.h</b>	4118
src/main/activemq/commands/	<b>JournalTopicAck.h</b>	4119
src/main/activemq/commands/	<b>JournalTrace.h</b>	4120
src/main/activemq/commands/	<b>JournalTransaction.h</b>	4121
src/main/activemq/commands/	<b>KeepAliveInfo.h</b>	4122
src/main/activemq/commands/	<b>LastPartialCommand.h</b>	4123
src/main/activemq/commands/	<b>LocalTransactionId.h</b>	4124
src/main/activemq/commands/	<b>Message.h</b>	4125
src/main/activemq/commands/	<b>MessageAck.h</b>	4127
src/main/activemq/commands/	<b>MessageDispatch.h</b>	4128
src/main/activemq/commands/	<b>MessageDispatchNotification.h</b>	4129
src/main/activemq/commands/	<b>MessageId.h</b>	4130
src/main/activemq/commands/	<b>MessagePull.h</b>	4131
src/main/activemq/commands/	<b>NetworkBridgeFilter.h</b>	4132
src/main/activemq/commands/	<b>PartialCommand.h</b>	4133
src/main/activemq/commands/	<b>ProducerAck.h</b>	4134
src/main/activemq/commands/	<b>ProducerId.h</b>	4135
src/main/activemq/commands/	<b>ProducerInfo.h</b>	4136
src/main/activemq/commands/	<b>RemoveInfo.h</b>	4137
src/main/activemq/commands/	<b>RemoveSubscriptionInfo.h</b>	4138
src/main/activemq/commands/	<b>ReplayCommand.h</b>	4139
src/main/activemq/commands/	<b>Response.h</b>	4140
src/main/activemq/commands/	<b>SessionId.h</b>	4141
src/main/activemq/commands/	<b>SessionInfo.h</b>	4142
src/main/activemq/commands/	<b>ShutdownInfo.h</b>	4143
src/main/activemq/commands/	<b>SubscriptionInfo.h</b>	4144
src/main/activemq/commands/	<b>TransactionId.h</b>	4145
src/main/activemq/commands/	<b>TransactionInfo.h</b>	4146
src/main/activemq/commands/	<b>WireFormatInfo.h</b>	4147
src/main/activemq/commands/	<b>XATransactionId.h</b>	4148
src/main/activemq/core/	<b>ActiveMQAckHandler.h</b>	4149
src/main/activemq/core/	<b>ActiveMQConnection.h</b>	4150
src/main/activemq/core/	<b>ActiveMQConnectionFactory.h</b>	4151
src/main/activemq/core/	<b>ActiveMQConnectionMetaData.h</b>	4152
src/main/activemq/core/	<b>ActiveMQConstants.h</b>	4153
src/main/activemq/core/	<b>ActiveMQConsumer.h</b>	4154
src/main/activemq/core/	<b>ActiveMQProducer.h</b>	4155

src/main/activemq/core/ActiveMQQueueBrowser.h	4156
src/main/activemq/core/ActiveMQSession.h	4157
src/main/activemq/core/ActiveMQSessionExecutor.h	4158
src/main/activemq/core/ActiveMQTransactionContext.h	4159
src/main/activemq/core/DispatchData.h	4160
src/main/activemq/core/Dispatcher.h	4161
src/main/activemq/core/MessageDispatchChannel.h	4162
src/main/activemq/core/PrefetchPolicy.h	4165
src/main/activemq/core/RedeliveryPolicy.h	4166
src/main/activemq/core/Synchronization.h	4167
src/main/activemq/core/policies/DefaultPrefetchPolicy.h	4163
src/main/activemq/core/policies/DefaultRedeliveryPolicy.h	4164
src/main/activemq/exceptions/ActiveMQException.h	4168
src/main/activemq/exceptions/BrokerException.h	4169
src/main/activemq/exceptions/ExceptionDefines.h	4170
src/main/activemq/io/LoggingInputStream.h	4175
src/main/activemq/io/LoggingOutputStream.h	4176
src/main/activemq/library/ActiveMQCPP.h	4177
src/main/activemq/state/CommandVisitor.h	4178
src/main/activemq/state/CommandVisitorAdapter.h	4179
src/main/activemq/state/ConnectionState.h	4181
src/main/activemq/state/ConnectionStateTracker.h	4182
src/main/activemq/state/ConsumerState.h	4183
src/main/activemq/state/ProducerState.h	4184
src/main/activemq/state/SessionState.h	4185
src/main/activemq/state/Tracked.h	4186
src/main/activemq/state/TransactionState.h	4187
src/main/activemq/threads/CompositeTask.h	4188
src/main/activemq/threads/CompositeTaskRunner.h	4189
src/main/activemq/threads/DedicatedTaskRunner.h	4190
src/main/activemq/threads/Task.h	4191
src/main/activemq/threads/TaskRunner.h	4192
src/main/activemq/transport/AbstractTransportFactory.h	4193
src/main/activemq/transport/CompositeTransport.h	4194
src/main/activemq/transport/DefaultTransportListener.h	4197
src/main/activemq/transport/IOTransport.h	4208
src/main/activemq/transport/Transport.h	4218
src/main/activemq/transport/TransportFactory.h	4219
src/main/activemq/transport/TransportFilter.h	4220
src/main/activemq/transport/TransportListener.h	4221
src/main/activemq/transport/TransportRegistry.h	4222
src/main/activemq/transport/correlator/FutureResponse.h	4195
src/main/activemq/transport/correlator/ResponseCorrelator.h	4196
src/main/activemq/transport/failover/BackupTransport.h	4198
src/main/activemq/transport/failover/BackupTransportPool.h	4199
src/main/activemq/transport/failover/CloseTransportsTask.h	4200
src/main/activemq/transport/failover/FailoverTransport.h	4201
src/main/activemq/transport/failover/FailoverTransportFactory.h	4202
src/main/activemq/transport/failover/FailoverTransportListener.h	4203
src/main/activemq/transport/failover/URIPool.h	4204
src/main/activemq/transport/inactivity/InactivityMonitor.h	4205
src/main/activemq/transport/inactivity/ReadChecker.h	4206
src/main/activemq/transport/inactivity/WriteChecker.h	4207
src/main/activemq/transport/logging/LoggingTransport.h	4209

src/main/activemq/transport/mock/ <b>InternalCommandListener.h</b>	4210
src/main/activemq/transport/mock/ <b>MockTransport.h</b>	4211
src/main/activemq/transport/mock/ <b>MockTransportFactory.h</b>	4212
src/main/activemq/transport/mock/ <b>ResponseBuilder.h</b>	4213
src/main/activemq/transport/tcp/ <b>SslTransport.h</b>	4214
src/main/activemq/transport/tcp/ <b>SslTransportFactory.h</b>	4215
src/main/activemq/transport/tcp/ <b>TcpTransport.h</b>	4216
src/main/activemq/transport/tcp/ <b>TcpTransportFactory.h</b>	4217
src/main/activemq/util/ <b>ActiveMQProperties.h</b>	4223
src/main/activemq/util/ <b>CMSExceptionSupport.h</b>	4224
src/main/activemq/util/ <b>CompositeData.h</b>	4226
src/main/activemq/util/ <b>Config.h</b>	4227
src/main/activemq/util/ <b>IdGenerator.h</b>	4230
src/main/activemq/util/ <b>LongSequenceGenerator.h</b>	4231
src/main/activemq/util/ <b>MarshallingSupport.h</b>	4232
src/main/activemq/util/ <b>MemoryUsage.h</b>	4233
src/main/activemq/util/ <b>PrimitiveList.h</b>	4234
src/main/activemq/util/ <b>PrimitiveMap.h</b>	4235
src/main/activemq/util/ <b>PrimitiveValueConverter.h</b>	4236
src/main/activemq/util/ <b>PrimitiveValueNode.h</b>	4237
src/main/activemq/util/ <b>URISupport.h</b>	4238
src/main/activemq/util/ <b>Usage.h</b>	4239
src/main/activemq/wireformat/ <b>MarshalAware.h</b>	4240
src/main/activemq/wireformat/ <b>WireFormat.h</b>	4628
src/main/activemq/wireformat/ <b>WireFormatFactory.h</b>	4629
src/main/activemq/wireformat/ <b>WireFormatNegotiator.h</b>	4630
src/main/activemq/wireformat/ <b>WireFormatRegistry.h</b>	4631
src/main/activemq/wireformat/openwire/ <b>OpenWireFormat.h</b>	4616
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatFactory.h</b>	4617
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatNegotiator.h</b>	4618
src/main/activemq/wireformat/openwire/ <b>OpenWireResponseBuilder.h</b>	4619
src/main/activemq/wireformat/openwire/marshal/ <b>BaseDataStreamMarshaller.h</b>	4241
src/main/activemq/wireformat/openwire/marshal/ <b>DataStreamMarshaller.h</b>	4242
src/main/activemq/wireformat/openwire/marshal/ <b>PrimitiveTypesMarshaller.h</b>	4243
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQBlobMessageMarshaller.h</b>	4244
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQBytesMessageMarshaller.h</b>	4250
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQDestinationMarshaller.h</b>	4256
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQMapMessageMarshaller.h</b>	4262
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQMessageMarshaller.h</b>	4268
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQObjectMessageMarshaller.h</b>	4274
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQQueueMarshaller.h</b>	4280
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQStreamMessageMarshaller.h</b>	4286
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQTempDestinationMarshaller.h</b>	4292
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQTempQueueMarshaller.h</b>	4298



src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
4304	
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
4310	
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	4316
src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	4322
src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h . . .	4328
src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h . . .	4334
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
4340	
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
4346	
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h .	4352
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	4358
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
4364	
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h . .	4370
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h .	4376
src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
4382	
src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
4388	
src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h .	4394
src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h	4400
src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h	4406
src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h	
4412	
src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	4418
src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
4424	
src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
4430	
src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
4436	
src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h . .	4442
src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
4448	
src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h .	4454
src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
4460	
src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
4466	
src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h . . . . .	4472
src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h . .	4478
src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h	
4484	
src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h	
4490	
src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h . . .	4496
src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h . . . . .	4502
src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h . .	4508
src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h	
4514	

src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h	
4520	
src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	4526
src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	4532
src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	4538
src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	4544
src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
4550	
src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h	
4556	
src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h	4562
src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h	4568
src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h	4574
src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	4580
src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
4586	
src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	4592
src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	4598
src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h	4604
src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h	
4610	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h	
4245	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h	
4251	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h	
4257	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
4263	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
4269	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
4275	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
4281	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
4287	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
4293	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
4299	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
4305	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
4311	
src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	4317
src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	4323
src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	4329
src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h	4335
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
4341	
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
4347	
src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	4353

src/main/activemq/wireformat/openwire/marshall/v2/	<b>ConnectionInfoMarshaller.h</b>	4359
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ConsumerControlMarshaller.h</b>	4365
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ConsumerIdMarshaller.h</b>	4371
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ConsumerInfoMarshaller.h</b>	4377
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ControlCommandMarshaller.h</b>	4383
src/main/activemq/wireformat/openwire/marshall/v2/	<b>DataArrayResponseMarshaller.h</b>	4389
src/main/activemq/wireformat/openwire/marshall/v2/	<b>DataResponseMarshaller.h</b>	4395
src/main/activemq/wireformat/openwire/marshall/v2/	<b>DestinationInfoMarshaller.h</b>	4401
src/main/activemq/wireformat/openwire/marshall/v2/	<b>DiscoveryEventMarshaller.h</b>	4407
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ExceptionResponseMarshaller.h</b>	4413
src/main/activemq/wireformat/openwire/marshall/v2/	<b>FlushCommandMarshaller.h</b>	4419
src/main/activemq/wireformat/openwire/marshall/v2/	<b>IntegerResponseMarshaller.h</b>	4425
src/main/activemq/wireformat/openwire/marshall/v2/	<b>JournalQueueAckMarshaller.h</b>	4431
src/main/activemq/wireformat/openwire/marshall/v2/	<b>JournalTopicAckMarshaller.h</b>	4437
src/main/activemq/wireformat/openwire/marshall/v2/	<b>JournalTraceMarshaller.h</b>	4443
src/main/activemq/wireformat/openwire/marshall/v2/	<b>JournalTransactionMarshaller.h</b>	4449
src/main/activemq/wireformat/openwire/marshall/v2/	<b>KeepAliveInfoMarshaller.h</b>	4455
src/main/activemq/wireformat/openwire/marshall/v2/	<b>LastPartialCommandMarshaller.h</b>	4461
src/main/activemq/wireformat/openwire/marshall/v2/	<b>LocalTransactionIdMarshaller.h</b>	4467
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MarshallerFactory.h</b>	4473
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessageAckMarshaller.h</b>	4479
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessageDispatchMarshaller.h</b>	4485
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessageDispatchNotificationMarshaller.h</b>	4491
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessageIdMarshaller.h</b>	4497
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessageMarshaller.h</b>	4503
src/main/activemq/wireformat/openwire/marshall/v2/	<b>MessagePullMarshaller.h</b>	4509
src/main/activemq/wireformat/openwire/marshall/v2/	<b>NetworkBridgeFilterMarshaller.h</b>	4515
src/main/activemq/wireformat/openwire/marshall/v2/	<b>PartialCommandMarshaller.h</b>	4521
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ProducerAckMarshaller.h</b>	4527
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ProducerIdMarshaller.h</b>	4533
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ProducerInfoMarshaller.h</b>	4539
src/main/activemq/wireformat/openwire/marshall/v2/	<b>RemoveInfoMarshaller.h</b>	4545
src/main/activemq/wireformat/openwire/marshall/v2/	<b>RemoveSubscriptionInfoMarshaller.h</b>	4551
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ReplayCommandMarshaller.h</b>	4557
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ResponseMarshaller.h</b>	4563
src/main/activemq/wireformat/openwire/marshall/v2/	<b>SessionIdMarshaller.h</b>	4569
src/main/activemq/wireformat/openwire/marshall/v2/	<b>SessionInfoMarshaller.h</b>	4575
src/main/activemq/wireformat/openwire/marshall/v2/	<b>ShutdownInfoMarshaller.h</b>	4581

src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**  
4587  
src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h** . 4593  
src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h** 4599  
src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h** 4605  
src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**  
4611  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**  
4246  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBytesMessageMarshaller.h**  
4252  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQDestinationMarshaller.h**  
4258  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**  
4264  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**  
4270  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQObjectMessageMarshaller.h**  
4276  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**  
4282  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**  
4288  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**  
4294  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**  
4300  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**  
4306  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**  
4312  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h** 4318  
src/main/activemq/wireformat/openwire/marshal/v3/**BaseCommandMarshaller.h** 4324  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h** . . . 4330  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h** . . . 4336  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**  
4342  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**  
4348  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h** . 4354  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h** 4360  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**  
4366  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h** . . 4372  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h** . 4378  
src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**  
4384  
src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**  
4390  
src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h** . 4396  
src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h** 4402  
src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h** 4408  
src/main/activemq/wireformat/openwire/marshal/v3/**ExceptionResponseMarshaller.h**  
4414

src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	4420
src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	4426
src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	4432
src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	4438
src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	4444
src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	4450
src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	4456
src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	4462
src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	4468
src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	4474
src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h	4480
src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h	4486
src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h	4492
src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	4498
src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	4504
src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h	4510
src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h	4516
src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	4522
src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	4528
src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	4534
src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	4540
src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	4546
src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	4552
src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h	4558
src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h	4564
src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h	4570
src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	4576
src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	4582
src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	4588
src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	4594
src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	4600
src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h	4606
src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h	4612
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h	4247
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h	4253
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	4259

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMapMessageMarshaller.h**  
4265  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMessageMarshaller.h**  
4271  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQObjectMessageMarshaller.h**  
4277  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQQueueMarshaller.h**  
4283  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQStreamMessageMarshaller.h**  
4289  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**  
4295  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempQueueMarshaller.h**  
4301  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempTopicMarshaller.h**  
4307  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTextMessageMarshaller.h**  
4313  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTopicMarshaller.h** 4319  
src/main/activemq/wireformat/openwire/marshal/v4/**BaseCommandMarshaller.h** 4325  
src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h** . . . 4331  
src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h** . . . 4337  
src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionControlMarshaller.h**  
4343  
src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionErrorMarshaller.h**  
4349  
src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionIdMarshaller.h** . 4355  
src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionInfoMarshaller.h** 4361  
src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerControlMarshaller.h**  
4367  
src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerIdMarshaller.h** . . 4373  
src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerInfoMarshaller.h** . 4379  
src/main/activemq/wireformat/openwire/marshal/v4/**ControlCommandMarshaller.h**  
4385  
src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**  
4391  
src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h** . 4397  
src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h** 4403  
src/main/activemq/wireformat/openwire/marshal/v4/**DiscoveryEventMarshaller.h** 4409  
src/main/activemq/wireformat/openwire/marshal/v4/**ExceptionResponseMarshaller.h**  
4415  
src/main/activemq/wireformat/openwire/marshal/v4/**FlushCommandMarshaller.h** 4421  
src/main/activemq/wireformat/openwire/marshal/v4/**IntegerResponseMarshaller.h**  
4427  
src/main/activemq/wireformat/openwire/marshal/v4/**JournalQueueAckMarshaller.h**  
4433  
src/main/activemq/wireformat/openwire/marshal/v4/**JournalTopicAckMarshaller.h**  
4439  
src/main/activemq/wireformat/openwire/marshal/v4/**JournalTraceMarshaller.h** . . 4445  
src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**  
4451  
src/main/activemq/wireformat/openwire/marshal/v4/**KeepAliveInfoMarshaller.h** . 4457  
src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**  
4463

src/main/activemq/wireformat/openwire/marshall/v4/LocalTransactionIdMarshaller.h	
4469	
src/main/activemq/wireformat/openwire/marshall/v4/MarshallerFactory.h . . . . .	4475
src/main/activemq/wireformat/openwire/marshall/v4/MessageAckMarshaller.h . .	4481
src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchMarshaller.h	
4487	
src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchNotificationMarshaller.h	
4493	
src/main/activemq/wireformat/openwire/marshall/v4/MessageIdMarshaller.h . . .	4499
src/main/activemq/wireformat/openwire/marshall/v4/MessageMarshaller.h . . . . .	4505
src/main/activemq/wireformat/openwire/marshall/v4/MessagePullMarshaller.h . .	4511
src/main/activemq/wireformat/openwire/marshall/v4/NetworkBridgeFilterMarshaller.h	
4517	
src/main/activemq/wireformat/openwire/marshall/v4/PartialCommandMarshaller.h	
4523	
src/main/activemq/wireformat/openwire/marshall/v4/ProducerAckMarshaller.h . .	4529
src/main/activemq/wireformat/openwire/marshall/v4/ProducerIdMarshaller.h . . .	4535
src/main/activemq/wireformat/openwire/marshall/v4/ProducerInfoMarshaller.h . .	4541
src/main/activemq/wireformat/openwire/marshall/v4/RemoveInfoMarshaller.h . .	4547
src/main/activemq/wireformat/openwire/marshall/v4/RemoveSubscriptionInfoMarshaller.h	
4553	
src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	
4559	
src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h . . . .	4565
src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h . . . .	4571
src/main/activemq/wireformat/openwire/marshall/v4/SessionInfoMarshaller.h . . .	4577
src/main/activemq/wireformat/openwire/marshall/v4/ShutdownInfoMarshaller.h .	4583
src/main/activemq/wireformat/openwire/marshall/v4/SubscriptionInfoMarshaller.h	
4589	
src/main/activemq/wireformat/openwire/marshall/v4/TransactionIdMarshaller.h .	4595
src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h	4601
src/main/activemq/wireformat/openwire/marshall/v4/WireFormatInfoMarshaller.h	4607
src/main/activemq/wireformat/openwire/marshall/v4/XATransactionIdMarshaller.h	
4613	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBlobMessageMarshaller.h	
4248	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBytesMessageMarshaller.h	
4254	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQDestinationMarshaller.h	
4260	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMapMessageMarshaller.h	
4266	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMessageMarshaller.h	
4272	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQObjectMessageMarshaller.h	
4278	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQQueueMarshaller.h	
4284	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStreamMessageMarshaller.h	
4290	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempDestinationMarshaller.h	
4296	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempQueueMarshaller.h	
4302	

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
4308	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
4314	
src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	4320
src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	4326
src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h . . .	4332
src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h . . .	4338
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
4344	
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
4350	
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h .	4356
src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	4362
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
4368	
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h . .	4374
src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h .	4380
src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
4386	
src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
4392	
src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h .	4398
src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h	4404
src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h	4410
src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
4416	
src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	4422
src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
4428	
src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
4434	
src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
4440	
src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h . .	4446
src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
4452	
src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h .	4458
src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
4464	
src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
4470	
src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h . . . .	4476
src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h . .	4482
src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h	
4488	
src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h	
4494	
src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h . . .	4500
src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h . . . .	4506
src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h . .	4512
src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h	
4518	



src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
4524	
src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	4530
src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	4536
src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	4542
src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h	4548
src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h	4554
src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h	4560
src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h	4566
src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h	4572
src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	4578
src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	4584
src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	4590
src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	4596
src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h	4602
src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h	4608
src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h	4614
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h	4249
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h	4255
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h	4261
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h	4267
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h	4273
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h	4279
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h	4285
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h	4291
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h	4297
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h	4303
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h	4309
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h	4315
src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h	4321
src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h	4327
src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h	4333
src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h	4339
src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h	4345
src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h	4351
src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h	4357

src/main/activemq/wireformat/openwire/marshal/v6/	<b>ConnectionInfoMarshaller.h</b>	4363
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ConsumerControlMarshaller.h</b>	4369
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ConsumerIdMarshaller.h</b>	4375
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ConsumerInfoMarshaller.h</b>	4381
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ControlCommandMarshaller.h</b>	4387
src/main/activemq/wireformat/openwire/marshal/v6/	<b>DataArrayResponseMarshaller.h</b>	4393
src/main/activemq/wireformat/openwire/marshal/v6/	<b>DataResponseMarshaller.h</b>	4399
src/main/activemq/wireformat/openwire/marshal/v6/	<b>DestinationInfoMarshaller.h</b>	4405
src/main/activemq/wireformat/openwire/marshal/v6/	<b>DiscoveryEventMarshaller.h</b>	4411
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ExceptionResponseMarshaller.h</b>	4417
src/main/activemq/wireformat/openwire/marshal/v6/	<b>FlushCommandMarshaller.h</b>	4423
src/main/activemq/wireformat/openwire/marshal/v6/	<b>IntegerResponseMarshaller.h</b>	4429
src/main/activemq/wireformat/openwire/marshal/v6/	<b>JournalQueueAckMarshaller.h</b>	4435
src/main/activemq/wireformat/openwire/marshal/v6/	<b>JournalTopicAckMarshaller.h</b>	4441
src/main/activemq/wireformat/openwire/marshal/v6/	<b>JournalTraceMarshaller.h</b>	4447
src/main/activemq/wireformat/openwire/marshal/v6/	<b>JournalTransactionMarshaller.h</b>	4453
src/main/activemq/wireformat/openwire/marshal/v6/	<b>KeepAliveInfoMarshaller.h</b>	4459
src/main/activemq/wireformat/openwire/marshal/v6/	<b>LastPartialCommandMarshaller.h</b>	4465
src/main/activemq/wireformat/openwire/marshal/v6/	<b>LocalTransactionIdMarshaller.h</b>	4471
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MarshallerFactory.h</b>	4477
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessageAckMarshaller.h</b>	4483
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessageDispatchMarshaller.h</b>	4489
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessageDispatchNotificationMarshaller.h</b>	4495
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessageIdMarshaller.h</b>	4501
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessageMarshaller.h</b>	4507
src/main/activemq/wireformat/openwire/marshal/v6/	<b>MessagePullMarshaller.h</b>	4513
src/main/activemq/wireformat/openwire/marshal/v6/	<b>NetworkBridgeFilterMarshaller.h</b>	4519
src/main/activemq/wireformat/openwire/marshal/v6/	<b>PartialCommandMarshaller.h</b>	4525
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ProducerAckMarshaller.h</b>	4531
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ProducerIdMarshaller.h</b>	4537
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ProducerInfoMarshaller.h</b>	4543
src/main/activemq/wireformat/openwire/marshal/v6/	<b>RemoveInfoMarshaller.h</b>	4549
src/main/activemq/wireformat/openwire/marshal/v6/	<b>RemoveSubscriptionInfoMarshaller.h</b>	4555
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ReplayCommandMarshaller.h</b>	4561
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ResponseMarshaller.h</b>	4567
src/main/activemq/wireformat/openwire/marshal/v6/	<b>SessionIdMarshaller.h</b>	4573
src/main/activemq/wireformat/openwire/marshal/v6/	<b>SessionInfoMarshaller.h</b>	4579
src/main/activemq/wireformat/openwire/marshal/v6/	<b>ShutdownInfoMarshaller.h</b>	4585

src/main/activemq/wireformat/openwire/marshall/v6/SubscriptionInfoMarshaller.h	
4591	
src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller.h	4597
src/main/activemq/wireformat/openwire/marshall/v6/TransactionInfoMarshaller.h	4603
src/main/activemq/wireformat/openwire/marshall/v6/WireFormatInfoMarshaller.h	4609
src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller.h	4615
src/main/activemq/wireformat/openwire/utils/BooleanStream.h	4620
src/main/activemq/wireformat/openwire/utils/HexTable.h	4621
src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h	4622
src/main/activemq/wireformat/stomp/StompCommandConstants.h	4623
src/main/activemq/wireformat/stomp/StompFrame.h	4624
src/main/activemq/wireformat/stomp/StompHelper.h	4625
src/main/activemq/wireformat/stomp/StompWireFormat.h	4626
src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	4627
src/main/cms/BytesMessage.h	4632
src/main/cms/Closeable.h	4633
src/main/cms/CMSException.h	4635
src/main/cms/CMSProperties.h	4636
src/main/cms/CMSSecurityException.h	4637
src/main/cms/Config.h	4228
src/main/cms/Connection.h	4638
src/main/cms/ConnectionFactory.h	4639
src/main/cms/ConnectionMetaData.h	4640
src/main/cms/DeliveryMode.h	4641
src/main/cms/Destination.h	4642
src/main/cms/ExceptionListener.h	4643
src/main/cms/IllegalStateException.h	4644
src/main/cms/InvalidClientIdException.h	4646
src/main/cms/InvalidDestinationException.h	4647
src/main/cms/InvalidSelectorException.h	4648
src/main/cms/MapMessage.h	4649
src/main/cms/Message.h	4126
src/main/cms/MessageConsumer.h	4650
src/main/cms/MessageEnumeration.h	4651
src/main/cms/MessageEOFException.h	4652
src/main/cms/MessageFormatException.h	4653
src/main/cms/MessageListener.h	4654
src/main/cms/MessageNotReadableException.h	4655
src/main/cms/MessageNotWriteableException.h	4656
src/main/cms/MessageProducer.h	4657
src/main/cms/ObjectMessage.h	4658
src/main/cms/Queue.h	4659
src/main/cms/QueueBrowser.h	4661
src/main/cms/Session.h	4662
src/main/cms/Startable.h	4663
src/main/cms/Stopable.h	4664
src/main/cms/StreamMessage.h	4665
src/main/cms/TemporaryQueue.h	4666
src/main/cms/TemporaryTopic.h	4667
src/main/cms/TextMessage.h	4668
src/main/cms/Topic.h	4669
src/main/cms/UnsupportedOperationException.h	4670
src/main/decaf/internal/AprPool.h	4672

src/main/decaf/internal/ <b>DecafRuntime.h</b> . . . . .	4673
src/main/decaf/internal/io/ <b>StandardErrorOutputStream.h</b> . . . . .	4674
src/main/decaf/internal/io/ <b>StandardInputStream.h</b> . . . . .	4675
src/main/decaf/internal/io/ <b>StandardOutputStream.h</b> . . . . .	4676
src/main/decaf/internal/net/ <b>DefaultServerSocketFactory.h</b> . . . . .	4677
src/main/decaf/internal/net/ <b>DefaultSocketFactory.h</b> . . . . .	4678
src/main/decaf/internal/net/ <b>Network.h</b> . . . . .	4679
src/main/decaf/internal/net/ <b>SocketFileDescriptor.h</b> . . . . .	4680
src/main/decaf/internal/net/ <b>URIEncoderDecoder.h</b> . . . . .	4696
src/main/decaf/internal/net/ <b>URIHelper.h</b> . . . . .	4697
src/main/decaf/internal/net/ <b>URIType.h</b> . . . . .	4698
src/main/decaf/internal/net/ssl/ <b>DefaultSSLContext.h</b> . . . . .	4681
src/main/decaf/internal/net/ssl/ <b>DefaultSSLServerSocketFactory.h</b> . . . . .	4682
src/main/decaf/internal/net/ssl/ <b>DefaultSSLSocketFactory.h</b> . . . . .	4683
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLContextSpi.h</b> . . . . .	4684
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLParameters.h</b> . . . . .	4685
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLServerSocket.h</b> . . . . .	4686
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLServerSocketFactory.h</b> . . . . .	4687
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLSocket.h</b> . . . . .	4688
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLSocketException.h</b> . . . . .	4689
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLSocketFactory.h</b> . . . . .	4690
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLSocketInputStream.h</b> . . . . .	4691
src/main/decaf/internal/net/ssl/openssl/ <b>OpenSSLSocketOutputStream.h</b> . . . . .	4692
src/main/decaf/internal/net/tcp/ <b>TcpSocket.h</b> . . . . .	4693
src/main/decaf/internal/net/tcp/ <b>TcpSocketInputStream.h</b> . . . . .	4694
src/main/decaf/internal/net/tcp/ <b>TcpSocketOutputStream.h</b> . . . . .	4695
src/main/decaf/internal/nio/ <b>BufferFactory.h</b> . . . . .	4699
src/main/decaf/internal/nio/ <b>ByteBuffer.h</b> . . . . .	4700
src/main/decaf/internal/nio/ <b>CharArrayBuffer.h</b> . . . . .	4701
src/main/decaf/internal/nio/ <b>DoubleArrayBuffer.h</b> . . . . .	4702
src/main/decaf/internal/nio/ <b>FloatArrayBuffer.h</b> . . . . .	4703
src/main/decaf/internal/nio/ <b>IntArrayBuffer.h</b> . . . . .	4704
src/main/decaf/internal/nio/ <b>LongArrayBuffer.h</b> . . . . .	4705
src/main/decaf/internal/nio/ <b>ShortArrayBuffer.h</b> . . . . .	4706
src/main/decaf/internal/security/unix/ <b>SecureRandomImpl.h</b> . . . . .	4707
src/main/decaf/internal/security/windows/ <b>SecureRandomImpl.h</b> . . . . .	4708
src/main/decaf/internal/util/ <b>ByteArrayAdapter.h</b> . . . . .	4709
src/main/decaf/internal/util/ <b>GenericResource.h</b> . . . . .	4720
src/main/decaf/internal/util/ <b>HexStringParser.h</b> . . . . .	4721
src/main/decaf/internal/util/ <b>Resource.h</b> . . . . .	4722
src/main/decaf/internal/util/ <b>ResourceLifecycleManager.h</b> . . . . .	4078
src/main/decaf/internal/util/ <b>TimerTaskHeap.h</b> . . . . .	4723
src/main/decaf/internal/util/concurrent/ <b>ConditionImpl.h</b> . . . . .	4710
src/main/decaf/internal/util/concurrent/ <b>MutexImpl.h</b> . . . . .	4711
src/main/decaf/internal/util/concurrent/ <b>SynchronizableImpl.h</b> . . . . .	4712
src/main/decaf/internal/util/concurrent/ <b>Transferer.h</b> . . . . .	4713
src/main/decaf/internal/util/concurrent/ <b>TransferQueue.h</b> . . . . .	4714
src/main/decaf/internal/util/concurrent/ <b>TransferStack.h</b> . . . . .	4715
src/main/decaf/internal/util/concurrent/unix/ <b>ConditionHandle.h</b> . . . . .	4716
src/main/decaf/internal/util/concurrent/unix/ <b>MutexHandle.h</b> . . . . .	4718
src/main/decaf/internal/util/concurrent/windows/ <b>ConditionHandle.h</b> . . . . .	4717
src/main/decaf/internal/util/concurrent/windows/ <b>MutexHandle.h</b> . . . . .	4719
src/main/decaf/internal/util/zip/ <b>crc32.h</b> . . . . .	4724
src/main/decaf/internal/util/zip/ <b>deflate.h</b> . . . . .	4725

src/main/decaf/internal/util/zip/gzguts.h	4728
src/main/decaf/internal/util/zip/inffast.h	4730
src/main/decaf/internal/util/zip/inffixed.h	4731
src/main/decaf/internal/util/zip/inflate.h	4732
src/main/decaf/internal/util/zip/inftrees.h	4734
src/main/decaf/internal/util/zip/trees.h	4735
src/main/decaf/internal/util/zip/zconf.h	4737
src/main/decaf/internal/util/zip/zlib.h	4739
src/main/decaf/internal/util/zip/zutil.h	4743
src/main/decaf/io/BlockingByteArrayInputStream.h	4746
src/main/decaf/io/BufferedInputStream.h	4747
src/main/decaf/io/BufferedOutputStream.h	4748
src/main/decaf/io/ByteArrayInputStream.h	4749
src/main/decaf/io/ByteArrayOutputStream.h	4750
src/main/decaf/io/Closeable.h	4634
src/main/decaf/io/DataInput.h	4751
src/main/decaf/io/DataInputStream.h	4752
src/main/decaf/io/DataOutput.h	4753
src/main/decaf/io/DataOutputStream.h	4754
src/main/decaf/io/EOFException.h	4755
src/main/decaf/io/FileDescriptor.h	4756
src/main/decaf/io/FilterInputStream.h	4757
src/main/decaf/io/FilterOutputStream.h	4758
src/main/decaf/io/Flushable.h	4759
src/main/decaf/io/InputStream.h	4760
src/main/decaf/io/InputStreamReader.h	4761
src/main/decaf/io/InterruptedIOException.h	4762
src/main/decaf/io/IOException.h	4763
src/main/decaf/io/OutputStream.h	4764
src/main/decaf/io/OutputStreamWriter.h	4765
src/main/decaf/io/PushbackInputStream.h	4766
src/main/decaf/io/Reader.h	4767
src/main/decaf/io/UnsupportedEncodingException.h	4768
src/main/decaf/io/UTFDataFormatException.h	4769
src/main/decaf/io/Writer.h	4770
src/main/decaf/lang/Appendable.h	4771
src/main/decaf/lang/ArrayPointer.h	4772
src/main/decaf/lang/Boolean.h	4774
src/main/decaf/lang/Byte.h	4775
src/main/decaf/lang/Character.h	4776
src/main/decaf/lang/CharSequence.h	4777
src/main/decaf/lang/Comparable.h	4778
src/main/decaf/lang/Double.h	4779
src/main/decaf/lang/Exception.h	4780
src/main/decaf/lang/Float.h	4792
src/main/decaf/lang/Integer.h	4793
src/main/decaf/lang/Iterable.h	4794
src/main/decaf/lang/Long.h	4795
src/main/decaf/lang/Math.h	4796
src/main/decaf/lang/Number.h	4797
src/main/decaf/lang/Pointer.h	4798
src/main/decaf/lang/Readable.h	4799
src/main/decaf/lang/Runnable.h	4800
src/main/decaf/lang/Runtime.h	4801

src/main/decaf/lang/	<b>Short.h</b>	4802
src/main/decaf/lang/	<b>String.h</b>	4803
src/main/decaf/lang/	<b>System.h</b>	4804
src/main/decaf/lang/	<b>Thread.h</b>	4805
src/main/decaf/lang/	<b>ThreadGroup.h</b>	4806
src/main/decaf/lang/	<b>Throwable.h</b>	4807
src/main/decaf/lang/exceptions/	<b>ClassCastException.h</b>	4781
src/main/decaf/lang/exceptions/	<b>ExceptionDefines.h</b>	4173
src/main/decaf/lang/exceptions/	<b>IllegalArgumentException.h</b>	4782
src/main/decaf/lang/exceptions/	<b>IllegalMonitorStateException.h</b>	4783
src/main/decaf/lang/exceptions/	<b>IllegalStateException.h</b>	4645
src/main/decaf/lang/exceptions/	<b>IllegalThreadStateException.h</b>	4784
src/main/decaf/lang/exceptions/	<b>IndexOutOfBoundsException.h</b>	4785
src/main/decaf/lang/exceptions/	<b>InterruptedException.h</b>	4786
src/main/decaf/lang/exceptions/	<b>InvalidStateException.h</b>	4787
src/main/decaf/lang/exceptions/	<b>NoSuchElementException.h</b>	4788
src/main/decaf/lang/exceptions/	<b>NullPointerException.h</b>	4789
src/main/decaf/lang/exceptions/	<b>NumberFormatException.h</b>	4790
src/main/decaf/lang/exceptions/	<b>RuntimeException.h</b>	4791
src/main/decaf/lang/exceptions/	<b>UnsupportedOperationException.h</b>	4671
src/main/decaf/net/	<b>BindException.h</b>	4808
src/main/decaf/net/	<b>ConnectException.h</b>	4809
src/main/decaf/net/	<b>HttpRetryException.h</b>	4810
src/main/decaf/net/	<b>Inet4Address.h</b>	4811
src/main/decaf/net/	<b>Inet6Address.h</b>	4812
src/main/decaf/net/	<b>InetAddress.h</b>	4813
src/main/decaf/net/	<b>InetSocketAddress.h</b>	4814
src/main/decaf/net/	<b>MalformedURLException.h</b>	4815
src/main/decaf/net/	<b>NoRouteToHostException.h</b>	4816
src/main/decaf/net/	<b>PortUnreachableException.h</b>	4817
src/main/decaf/net/	<b>ProtocolException.h</b>	4818
src/main/decaf/net/	<b>ServerSocket.h</b>	4819
src/main/decaf/net/	<b>ServerSocketFactory.h</b>	4820
src/main/decaf/net/	<b>Socket.h</b>	4821
src/main/decaf/net/	<b>SocketAddress.h</b>	4822
src/main/decaf/net/	<b>SocketError.h</b>	4823
src/main/decaf/net/	<b>SocketException.h</b>	4824
src/main/decaf/net/	<b>SocketFactory.h</b>	4825
src/main/decaf/net/	<b>SocketImpl.h</b>	4826
src/main/decaf/net/	<b>SocketImplFactory.h</b>	4827
src/main/decaf/net/	<b>SocketOptions.h</b>	4828
src/main/decaf/net/	<b>SocketTimeoutException.h</b>	4829
src/main/decaf/net/	<b>UnknownHostException.h</b>	4837
src/main/decaf/net/	<b>UnknownServiceException.h</b>	4838
src/main/decaf/net/	<b>URI.h</b>	4839
src/main/decaf/net/	<b>URISyntaxException.h</b>	4840
src/main/decaf/net/	<b>URL.h</b>	4841
src/main/decaf/net/	<b>URLDecoder.h</b>	4842
src/main/decaf/net/	<b>URLEncoder.h</b>	4843
src/main/decaf/net/ssl/	<b>SSLContext.h</b>	4830
src/main/decaf/net/ssl/	<b>SSLContextSpi.h</b>	4831
src/main/decaf/net/ssl/	<b>SSLParameters.h</b>	4832
src/main/decaf/net/ssl/	<b>SSLServerSocket.h</b>	4833
src/main/decaf/net/ssl/	<b>SSLServerSocketFactory.h</b>	4834

src/main/decaf/net/ssl/SSLSocket.h	4835
src/main/decaf/net/ssl/SSLSocketFactory.h	4836
src/main/decaf/nio/Buffer.h	4844
src/main/decaf/nio/BufferOverflowException.h	4845
src/main/decaf/nio/BufferUnderflowException.h	4846
src/main/decaf/nio/ByteBuffer.h	4847
src/main/decaf/nio/CharBuffer.h	4848
src/main/decaf/nio/DoubleBuffer.h	4849
src/main/decaf/nio/FloatBuffer.h	4850
src/main/decaf/nio/IntBuffer.h	4851
src/main/decaf/nio/InvalidMarkException.h	4852
src/main/decaf/nio/LongBuffer.h	4853
src/main/decaf/nio/ReadOnlyBufferException.h	4854
src/main/decaf/nio/ShortBuffer.h	4855
src/main/decaf/security/GeneralSecurityException.h	4864
src/main/decaf/security/InvalidKeyException.h	4865
src/main/decaf/security/Key.h	4866
src/main/decaf/security/KeyException.h	4867
src/main/decaf/security/KeyManagementException.h	4868
src/main/decaf/security/NoSuchAlgorithmException.h	4869
src/main/decaf/security/NoSuchProviderException.h	4870
src/main/decaf/security/Principal.h	4871
src/main/decaf/security/PublicKey.h	4872
src/main/decaf/security/SecureRandom.h	4873
src/main/decaf/security/SecureRandomSpi.h	4874
src/main/decaf/security/SignatureException.h	4875
src/main/decaf/security/auth/x500/X500Principal.h	4856
src/main/decaf/security/cert/Certificate.h	4857
src/main/decaf/security/cert/CertificateEncodingException.h	4858
src/main/decaf/security/cert/CertificateException.h	4859
src/main/decaf/security/cert/CertificateExpiredException.h	4860
src/main/decaf/security/cert/CertificateNotYetValidException.h	4861
src/main/decaf/security/cert/CertificateParsingException.h	4862
src/main/decaf/security/cert/X509Certificate.h	4863
src/main/decaf/util/AbstractCollection.h	4876
src/main/decaf/util/AbstractList.h	4877
src/main/decaf/util/AbstractMap.h	4878
src/main/decaf/util/AbstractQueue.h	4879
src/main/decaf/util/AbstractSequentialList.h	4880
src/main/decaf/util/AbstractSet.h	4881
src/main/decaf/util/Collection.h	4882
src/main/decaf/util/Comparator.h	4883
src/main/decaf/util/Config.h	4229
src/main/decaf/util/Date.h	4921
src/main/decaf/util/Iterator.h	4922
src/main/decaf/util/List.h	4923
src/main/decaf/util/ListIterator.h	4924
src/main/decaf/util/Map.h	4945
src/main/decaf/util/PriorityQueue.h	4946
src/main/decaf/util/Properties.h	4947
src/main/decaf/util/Queue.h	4660
src/main/decaf/util/Random.h	4948
src/main/decaf/util/Set.h	4949
src/main/decaf/util/StlList.h	4950

src/main/decaf/util/ <b>StlMap.h</b>	4951
src/main/decaf/util/ <b>StlQueue.h</b>	4952
src/main/decaf/util/ <b>StlSet.h</b>	4953
src/main/decaf/util/ <b>StringTokenizer.h</b>	4954
src/main/decaf/util/ <b>Timer.h</b>	4955
src/main/decaf/util/ <b>TimerTask.h</b>	4956
src/main/decaf/util/ <b>UUID.h</b>	4957
src/main/decaf/util/comparators/ <b>Less.h</b>	4884
src/main/decaf/util/concurrent/ <b>BlockingQueue.h</b>	4889
src/main/decaf/util/concurrent/ <b>BrokenBarrierException.h</b>	4890
src/main/decaf/util/concurrent/ <b>Callable.h</b>	4891
src/main/decaf/util/concurrent/ <b>CancellationException.h</b>	4892
src/main/decaf/util/concurrent/ <b>Concurrent.h</b>	4893
src/main/decaf/util/concurrent/ <b>ConcurrentMap.h</b>	4894
src/main/decaf/util/concurrent/ <b>ConcurrentStlMap.h</b>	4895
src/main/decaf/util/concurrent/ <b>CountDownLatch.h</b>	4896
src/main/decaf/util/concurrent/ <b>Delayed.h</b>	4897
src/main/decaf/util/concurrent/ <b>ExecutionException.h</b>	4898
src/main/decaf/util/concurrent/ <b>Executor.h</b>	4899
src/main/decaf/util/concurrent/ <b>ExecutorService.h</b>	4900
src/main/decaf/util/concurrent/ <b>Future.h</b>	4901
src/main/decaf/util/concurrent/ <b>Lock.h</b>	4902
src/main/decaf/util/concurrent/ <b>Mutex.h</b>	4908
src/main/decaf/util/concurrent/ <b>PooledThread.h</b>	4909
src/main/decaf/util/concurrent/ <b>PooledThreadListener.h</b>	4910
src/main/decaf/util/concurrent/ <b>RejectedExecutionException.h</b>	4911
src/main/decaf/util/concurrent/ <b>RejectedExecutionHandler.h</b>	4912
src/main/decaf/util/concurrent/ <b>Semaphore.h</b>	4913
src/main/decaf/util/concurrent/ <b>Synchronizable.h</b>	4914
src/main/decaf/util/concurrent/ <b>SynchronousQueue.h</b>	4915
src/main/decaf/util/concurrent/ <b>TaskListener.h</b>	4916
src/main/decaf/util/concurrent/ <b>ThreadFactory.h</b>	4917
src/main/decaf/util/concurrent/ <b>ThreadPool.h</b>	4918
src/main/decaf/util/concurrent/ <b>TimeoutException.h</b>	4919
src/main/decaf/util/concurrent/ <b>TimeUnit.h</b>	4920
src/main/decaf/util/concurrent/atomic/ <b>AtomicBoolean.h</b>	4885
src/main/decaf/util/concurrent/atomic/ <b>AtomicInteger.h</b>	4886
src/main/decaf/util/concurrent/atomic/ <b>AtomicReferenceCounter.h</b>	4887
src/main/decaf/util/concurrent/atomic/ <b>AtomicReference.h</b>	4888
src/main/decaf/util/concurrent/locks/ <b>Condition.h</b>	4904
src/main/decaf/util/concurrent/locks/ <b>Lock.h</b>	4903
src/main/decaf/util/concurrent/locks/ <b>LockSupport.h</b>	4905
src/main/decaf/util/concurrent/locks/ <b>ReadWriteLock.h</b>	4906
src/main/decaf/util/concurrent/locks/ <b>ReentrantLock.h</b>	4907
src/main/decaf/util/logging/ <b>ConsoleHandler.h</b>	4925
src/main/decaf/util/logging/ <b>ErrorManager.h</b>	4926
src/main/decaf/util/logging/ <b>Filter.h</b>	4927
src/main/decaf/util/logging/ <b>Formatter.h</b>	4928
src/main/decaf/util/logging/ <b>Handler.h</b>	4929
src/main/decaf/util/logging/ <b>Level.h</b>	4930
src/main/decaf/util/logging/ <b>Logger.h</b>	4931
src/main/decaf/util/logging/ <b>LoggerCommon.h</b>	4932
src/main/decaf/util/logging/ <b>LoggerDefines.h</b>	4933
src/main/decaf/util/logging/ <b>LoggerHierarchy.h</b>	4935



src/main/decaf/util/logging/ <b>LogManager.h</b> . . . . .	4936
src/main/decaf/util/logging/ <b>LogRecord.h</b> . . . . .	4937
src/main/decaf/util/logging/ <b>LogWriter.h</b> . . . . .	4938
src/main/decaf/util/logging/ <b>MarkBlockLogger.h</b> . . . . .	4939
src/main/decaf/util/logging/ <b>PropertiesChangeListener.h</b> . . . . .	4940
src/main/decaf/util/logging/ <b>SimpleFormatter.h</b> . . . . .	4941
src/main/decaf/util/logging/ <b>SimpleLogger.h</b> . . . . .	4942
src/main/decaf/util/logging/ <b>StreamHandler.h</b> . . . . .	4943
src/main/decaf/util/logging/ <b>XMLFormatter.h</b> . . . . .	4944
src/main/decaf/util/zip/ <b>Adler32.h</b> . . . . .	4958
src/main/decaf/util/zip/ <b>CheckedInputStream.h</b> . . . . .	4959
src/main/decaf/util/zip/ <b>CheckedOutputStream.h</b> . . . . .	4960
src/main/decaf/util/zip/ <b>Checksum.h</b> . . . . .	4961
src/main/decaf/util/zip/ <b>CRC32.h</b> . . . . .	4962
src/main/decaf/util/zip/ <b>DataFormatException.h</b> . . . . .	4963
src/main/decaf/util/zip/ <b>Deflater.h</b> . . . . .	4964
src/main/decaf/util/zip/ <b>DeflaterOutputStream.h</b> . . . . .	4965
src/main/decaf/util/zip/ <b>Inflater.h</b> . . . . .	4966
src/main/decaf/util/zip/ <b>InflaterInputStream.h</b> . . . . .	4967
src/main/decaf/util/zip/ <b>ZipException.h</b> . . . . .	4968



## Chapter 5

# Namespace Documentation

### 5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

#### Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

#### 5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.2 activemq::cmsutil Namespace Reference

### Data Structures

- class **CachedConsumer**  
*A cached message consumer contained within a pooled session.*
- class **CachedProducer**  
*A cached message producer contained within a pooled session.*
- class **CmsAccessor**  
*Base class for `activemq.cmsutil.CmsTemplate` (p. 1170) and other CMS-accessing gateway helpers, defining common properties such as the CMS `cms.ConnectionFactory` (p. 1324) to operate on.*
- class **CmsDestinationAccessor**  
*Extends the `CmsAccessor` (p. 1153) to add support for resolving destination names.*
- class **CmsTemplate**  
*`CmsTemplate` (p. 1170) simplifies performing synchronous CMS operations.*
- class **DestinationResolver**  
*Resolves a CMS destination name to a `Destination`.*
- class **DynamicDestinationResolver**  
*Resolves a CMS destination name to a `Destination`.*
- class **MessageCreator**  
*Creates the user-defined message to be sent by the `CmsTemplate` (p. 1170).*
- class **PooledSession**  
*A pooled session object that wraps around a delegate session.*
- class **ProducerCallback**  
*Callback for sending a message to a CMS destination.*
- class **ResourceLifecycleManager**  
*Manages the lifecycle of a set of CMS resources.*
- class **SessionCallback**  
*Callback for executing any number of operations on a provided CMS Session.*
- class **SessionPool**  
*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

## 5.3 activemq::commands Namespace Reference

### Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

*This class represents an Exception sent from the Broker.*

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**

- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

## 5.4 activemq::core Namespace Reference

### Namespaces

- namespace **policies**

### Data Structures

- class **ActiveMQAckHandler**

*Interface class that is used to give CMS Messages an interface to Ack themselves with.*

- class **ActiveMQConnection**

*Concrete connection used for all connectors to the ActiveMQ broker.*

- class **ActiveMQConnectionFactory**

- class **ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 273) class.*

- class **ActiveMQConstants**

*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.*

- class **ActiveMQConsumer**

- class **ActiveMQProducer**

- class **ActiveMQQueueBrowser**

- class **ActiveMQSession**

- class **ActiveMQSessionExecutor**

*Delegate dispatcher for a single session.*

- class **ActiveMQTransactionContext**

*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*

- class **DispatchData**

*Simple POCO that contains the information necessary to route a message to a specified consumer.*

- class **Dispatcher**

*Interface for an object responsible for dispatching messages to consumers.*

- class **MessageDispatchChannel**

- class **PrefetchPolicy**

*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*

- class **RedeliveryPolicy**

*Interface for a **RedeliveryPolicy** (p. 3177) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

- class **Synchronization**

*Transacted Object **Synchronization** (p. 3715), used to sync the events of a Transaction with the items in the Transaction.*



## 5.5 activemq::core::policies Namespace Reference

### Data Structures

- class **DefaultPrefetchPolicy**
- class **DefaultRedeliveryPolicy**

## 5.6 activemq::exceptions Namespace Reference

### Data Structures

- class **ActiveMQException**
- class **BrokerException**

## 5.7 activemq::io Namespace Reference

### Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**

*OutputStream filter that just logs the data being written.*

## 5.8 activemq::library Namespace Reference

### Data Structures

- class `ActiveMQCPP`

## 5.9 activemq::state Namespace Reference

### Data Structures

- class **CommandVisitor**

*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*

- class **CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1200) that returns NULL for all calls.*

- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

## 5.10 activemq::threads Namespace Reference

### Data Structures

- class **CompositeTask**

*Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p. 1223).*

- class **CompositeTaskRunner**

*A **Task** (p. 3734) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.*

- class **DedicatedTaskRunner**

- class **Task**

*Represents a unit of work that requires one or more iterations to complete.*

- class **TaskRunner**

## 5.11 activemq::transport Namespace Reference

### Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

### Data Structures

- class **AbstractTransportFactory**  
*Abstract implementation of the **TransportFactory** (p. 3889) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3889) instances.*
- class **CompositeTransport**  
*A Composite **Transport** (p. 3883) is a **Transport** (p. 3883) implementation that is composed of several Transports.*
- class **DefaultTransportListener**
- class **IOTransport**  
*Implementation of the **Transport** (p. 3883) interface that performs marshaling of **commands** (p. 87) to IO streams.*
- class **Transport**  
*Interface for a **transport** (p. 97) layer for command objects.*
- class **TransportFactory**  
*Defines the interface for Factories that create Transports or TransportFilters.*
- class **TransportFilter**  
*A filter on the **transport** (p. 97) layer.*
- class **TransportListener**  
*A listener of asynchronous **exceptions** (p. 92) from a command **transport** (p. 97) object.*
- class **TransportRegistry**  
*Registry of all **Transport** (p. 3883) Factories that are available to the client at runtime.*

## 5.12 activemq::transport::correlator Namespace Reference

### Data Structures

- class **FutureResponse**

*A container that holds a response object.*

- class **ResponseCorrelator**

*This type of **transport** (p. 97) filter is responsible for correlating asynchronous responses with requests.*



## 5.13 activemq::transport::failover Namespace Reference

### Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**

*Creates an instance of a **FailoverTransport** (p. 1873).*

- class **FailoverTransportListener**

*Utility class used by the **Transport** (p. 3883) to perform the work of responding to events from the active **Transport** (p. 3883).*

- class **URIPool**

## 5.14 activemq::transport::inactivity Namespace Reference

### Data Structures

- class **InactivityMonitor**
- class **ReadChecker**  
*Runnable class that is used by the {}.*
- class **WriteChecker**  
*Runnable class used by the {}.*

## 5.15 activemq::transport::logging Namespace Reference

### Data Structures

- class **LoggingTransport**

*A **transport** (p. 97) filter that logs **commands** (p. 87) as they are sent/received.*

## 5.16 activemq::transport::mock Namespace Reference

### Data Structures

- class **InternalCommandListener**

*Listens for Commands sent from the **MockTransport** (p. 2768).*

- class **MockTransport**

*The **MockTransport** (p. 2768) defines a base level **Transport** (p. 3883) class that is intended to be used in place of an a regular protocol **Transport** (p. 3883) such as TCP.*

- class **MockTransportFactory**

*Manufactures MockTransports, which are objects that read from input streams and write to output streams.*

- class **ResponseBuilder**

*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

## 5.17 activemq::transport::tcp Namespace Reference

### Data Structures

- class **SslTransport**

***Transport** (p. 3883) for connecting to a Broker using an SSL Socket.*

- class **SslTransportFactory**

- class **TcpTransport**

*Implements a TCP/IP based **transport** (p. 97) filter, this **transport** (p. 97) is meant to wrap an instance of an **IOTransport** (p. 2145).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 3752).*

## 5.18 activemq::util Namespace Reference

### Data Structures

- class **ActiveMQProperties**

*Implementation of the **CMSProperties** interface that delegates to a **decaf::util::Properties** (p. 3126) object.*

- class **CMSExceptionSupport**
- class **CompositeData**

*Represents a Composite URI.*

- class **IdGenerator**
- class **LongSequenceGenerator**

*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*

- class **MarshallingSupport**
- class **MemoryUsage**
- class **PrimitiveList**

*List of primitives.*

- class **PrimitiveMap**

*Map of named primitives.*

- class **PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3012) from one type to another.*

- class **PrimitiveValueNode**

*Class that wraps around a single value of one of the many types.*

- class **URISupport**
- class **Usage**

## 5.19 activemq::wireformat Namespace Reference

### Namespaces

- namespace **openwire**
- namespace **stomp**

### Data Structures

- class **MarshalAware**
- class **WireFormat**  
*Provides a mechanism to marshal **commands** (p. 87) into and out of packets or into and out of streams, Channels and Datagrams.*
- class **WireFormatFactory**  
*The **WireFormatFactory** (p. 3980) is the interface that all **WireFormatFactory** (p. 3980) classes must extend.*
- class **WireFormatNegotiator**  
*Defines a **WireFormatNegotiator** (p. 4016) which allows a **WireFormat** (p. 3976) to.*
- class **WireFormatRegistry**  
*Registry of all **WireFormat** (p. 3976) Factories that are available to the client at runtime.*

## 5.20 activemq::wireformat::openwire Namespace Reference

### Namespaces

- namespace **marshal**
- namespace **utils**

### Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**



## 5.21 activemq::wireformat::openwire::marshal Namespace Reference

### Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**
- namespace **v6**

### Data Structures

- class **BaseDataStreamMarshaller**  
*Base class for all Marshallers that **marshal** (p. 107) DataStructures to and from the wire using the OpenWire protocol.*
- class **DataStreamMarshaller**  
*Base class for all classes that **marshal** (p. 107) **commands** (p. 87) for Openwire.*
- class **PrimitiveTypesMarshaller**  
*This class wraps the functionality needed to **marshal** (p. 107) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

## 5.22 activemq::wireformat::openwire::marshal::v1 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 213).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 253).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQDestinationMarshaller (p. 337).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMapMessageMarshaller (p. 378).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 405).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 450).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQQueueMarshaller (p. 495).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 556).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTempDestinationMarshaller (p. 584).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTempQueueMarshaller (p. 613).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTempTopicMarshaller (p. 646).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 675).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 703).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 779).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 880).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 912).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1280).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1312).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1343).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1373).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1418).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1447).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1480).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1509).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1543).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1606).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1744).*

- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1778).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1862).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1956).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2110).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2179).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2208).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2231).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2262).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2289).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2324).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2371).*
- class **MarshallerFactory**  
*Used to createmarshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2581).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2622).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2651).*

- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2688).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2713).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2760).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2816).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2942).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3060).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3092).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3109).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3210).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3227).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3258).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3315).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3403).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3419).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3481).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3680).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3827).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3855).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4008).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4048).*

## 5.23 activemq:wireformat::openwire::marshal:v2 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 221).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 269).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 349).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 390).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 417).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 507).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 568).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 596).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 625).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 687).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 715).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 800).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 892).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 924).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1292).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1300).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1331).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1361).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1406).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1435).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1468).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1497).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1531).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1594).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1732).*



- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1766).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1846).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1944).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2098).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2163).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2192).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2215).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2246).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2273).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2312).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2355).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2569).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2606).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2639).*

- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2668).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2703).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2744).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2796).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2926).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3040).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3072).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3105).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3198).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3235).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3262).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3300).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3383).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3427).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3477).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3696).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3831).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3871).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4000).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4040).*

## 5.24 activemq::wireformat::openwire::marshal::v3 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 209).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 249).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 333).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 374).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 446).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 552).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 580).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 609).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 638).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 667).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 695).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 765).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 872).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 904).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1272).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1304).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1335).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1365).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1410).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1439).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1472).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1501).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1535).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1598).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1736).*

- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1770).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1850).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1948).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2102).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2171).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2196).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2219).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2250).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2277).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2308).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2359).*
- class **MarshallerFactory**  
*Used to createmarshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2573).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2610).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2643).*

- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2680).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2698).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2752).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2808).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2934).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3048).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3080).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3117).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3206).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3231).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3266).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3310).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3399).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3423).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3489).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3676).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3835).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3859).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4012).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4052).*



## 5.25 activemq:wireformat::openwire::marshal:v4 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 217).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 257).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 341).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 382).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 409).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 454).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 560).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 588).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 617).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 642).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 671).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 699).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 772).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 876).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 908).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1276).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1308).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1339).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1369).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1414).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1443).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1476).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1505).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1539).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1602).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1740).*

- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1774).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1858).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1952).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2106).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2175).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2204).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2227).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2258).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2281).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2320).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2367).*
- class **MarshallerFactory**  
*Used to createmarshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2577).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2618).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2647).*

- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2672).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2708).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2756).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2812).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2938).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3044).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3076).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3101).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3218).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3247).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3254).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3295).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3387).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3431).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3493).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3688).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3839).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3867).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4004).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4044).*

## 5.26 `activemq::wireformat::openwire::marshal::v5` Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 225).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 261).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 345).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 386).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 413).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 458).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 503).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 564).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 592).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 621).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 650).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 679).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 707).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 786).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 884).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 916).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1284).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1316).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1347).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1377).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1422).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1451).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1484).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1513).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1547).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1586).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1752).*

- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1782).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1854).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1960).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2114).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2167).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2188).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2235).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2254).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2285).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2316).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2363).*
- class **MarshallerFactory**  
*Used to createmarshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2585).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2614).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2655).*



- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2676).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2693).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2748).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2804).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2930).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3052).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3084).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3113).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3214).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3243).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3274).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3305).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3395).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3415).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3485).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3684).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3823).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3851).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3992).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4056).*

## 5.27 activemq:wireformat::openwire::marshal:v6 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 229).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 265).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 353).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 394).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 421).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 466).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 511).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 572).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 600).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 629).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 683).*

- class **ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 711).*
- class **BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 793).*
- class **BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 888).*
- class **BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 920).*
- class **ConnectionControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1288).*
- class **ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1320).*
- class **ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1351).*
- class **ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1381).*
- class **ConsumerControlMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1426).*
- class **ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1455).*
- class **ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1488).*
- class **ControlCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1517).*
- class **DataArrayResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1551).*
- class **DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1590).*
- class **DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1748).*

- class **DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1762).*
- class **ExceptionResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1842).*
- class **FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1940).*
- class **IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2094).*
- class **JournalQueueAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2159).*
- class **JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2200).*
- class **JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2223).*
- class **JournalTransactionMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2242).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2269).*
- class **LastPartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2304).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2351).*
- class **MarshallerFactory**  
*Used to createmarshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2565).*
- class **MessageDispatchMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchMarshaller** (p. 2626).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2635).*

- class **MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2684).*
- class **MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2718).*
- class **MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2764).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2800).*
- class **PartialCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2922).*
- class **ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3056).*
- class **ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3088).*
- class **ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3121).*
- class **RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3202).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3239).*
- class **ReplayCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3270).*
- class **ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3320).*
- class **SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3391).*
- class **SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3411).*
- class **ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3473).*
- class **SubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3692).*

- class **TransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3843).*

- class **TransactionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3863).*

- class **WireFormatInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3996).*

- class **XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4036).*

## 5.28 activemq::wireformat::openwire::utils Namespace Reference

### Data Structures

- class **BooleanStream**

*Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.*

- class **HexTable**

*The **HexTable** (p. 1984) class maps hexadecimal strings to the value of an index into the table, i.e.*

- class **MessagePropertyInterceptor**

*Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWireMessage` properties.*



## 5.29 activemq::wireformat::stomp Namespace Reference

### Data Structures

- class **StompCommandConstants**
- class **StompFrame**

*A Stomp-level message frame that encloses all messages to and from the broker.*

- class **StompHelper**

*Utility Methods used when marshaling to and from StompFrame's.*

- class **StompWireFormat**
- class **StompWireFormatFactory**

*Factory used to create the Stomp Wire Format instance.*

## 5.30 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Data Structures

- class **BytesMessage**  
*A **BytesMessage** (p. 1056) object is used to send a message containing a stream of unsigned bytes.*
- class **Closeable**  
*Interface for a class that implements the close method.*
- class **CMSException**  
*CMS API Exception that is the base for all exceptions thrown from CMS classes.*
- class **CMSProperties**  
*Interface for a Java-like properties object.*
- class **CMSSecurityException**  
*This exception must be thrown when a provider rejects a user name/password submitted by a client.*
- class **Connection**  
*The client's connection to its provider.*
- class **ConnectionFactory**  
*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1262) objects returned implement the CMS **Connection** (p. 1262) interface and hide the CMS Provider specific implementation details behind that interface.*
- class **ConnectionMetaData**  
*A **ConnectionMetaData** (p. 1385) object provides information describing the **Connection** (p. 1262) object.*
- class **DeliveryMode**  
*This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.*
- class **Destination**  
*A **Destination** (p. 1723) object encapsulates a provider-specific address.*
- class **ExceptionListener**  
*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1838) that is registered with the **Connection** (p. 1262).*
- class **IllegalStateException**  
*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*

- class **InvalidClientIdException**

*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*

- class **InvalidDestinationException**

*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*

- class **InvalidSelectorException**

*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*

- class **MapMessage**

*A **MapMessage** (p. 2472) object is used to send a set of name-value pairs.*

- class **Message**

*Root of all messages.*

- class **MessageConsumer**

*A client uses a **MessageConsumer** (p. 2589) to received messages from a destination.*

- class **MessageEnumeration**

*Defines an object that enumerates a collection of Messages.*

- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3652) or **BytesMessage** (p. 1056) is being read.*

- class **MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

- class **MessageListener**

*A **MessageListener** (p. 2692) object is used to receive asynchronously delivered messages.*

- class **MessageNotReadableException**

*This exception must be thrown when a CMS client attempts to read a write-only message.*

- class **MessageNotWriteableException**

*This exception must be thrown when a CMS client attempts to write to a read-only message.*

- class **MessageProducer**

*A client uses a **MessageProducer** (p. 2725) object to send messages to a **Destination** (p. 1723).*

- class **ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

- class **Queue**

*An interface encapsulating a provider-specific queue name.*

- class **QueueBrowser**  
*This class implements in interface for browsing the messages in a **Queue** (p. 3148) without removing them.*
- class **Session**  
*A **Session** (p. 3365) object is a single-threaded context for producing and consuming messages.*
- class **Startable**  
*Interface for a class that implements the start method.*
- class **Stoppable**  
*Interface for a class that implements the stop method.*
- class **StreamMessage**  
*Interface for a **StreamMessage** (p. 3652).*
- class **TemporaryQueue**  
*Defines a Temporary **Queue** (p. 3148) based **Destination** (p. 1723).*
- class **TemporaryTopic**  
*Defines a Temporary **Topic** (p. 3817) based **Destination** (p. 1723).*
- class **TextMessage**  
*Interface for a text message.*
- class **Topic**  
*An interface encapsulating a provider-specific topic name.*
- class **UnsupportedOperationException**  
*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

### 5.30.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.31 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

#### 5.31.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.32 decaf::internal Namespace Reference

### Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

### Data Structures

- class **AprPool**  
*Wraps an APR pool object so that classes in **decaf** (p. 143) can create a static member for use in static methods where apr function calls that need a pool are made.*
- class **DecafRuntime**  
*Handles APR initialization and termination.*

## 5.33 decaf::internal::io Namespace Reference

### Data Structures

- class **StandardErrorOutputStream**

*Wrapper Around the Standard error Output facility on the current platform.*

- class **StandardInputStream**
- class **StandardOutputStream**

## 5.34 decaf::internal::net Namespace Reference

### Namespaces

- namespace **ssl**
- namespace **tcp**

### Data Structures

- class **DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*
- class **DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*
- class **Network**  
*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*
- class **SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*
- class **URIEncoderDecoder**
- class **URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*
- class **URIType**  
*Basic type object that holds data that composes a given URI.*



## 5.35 decaf::internal::net::ssl Namespace Reference

### Namespaces

- namespace **openssl**

### Data Structures

- class **DefaultSSLContext**  
*Default SSL Context manager for the Decaf library.*
- class **DefaultSSLServerSocketFactory**  
*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*
- class **DefaultSSLSocketFactory**  
*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

## 5.36 decaf::internal::net::ssl::openssl Namespace Reference

### Data Structures

- class **OpenSSLContextSpi**  
*Provides an SSLContext that wraps the OpenSSL API.*
- class **OpenSSLParameters**  
*Container class for parameters that are Common to OpenSSL socket classes.*
- class **OpenSSLServerSocket**  
*SSLServerSocket based on OpenSSL library `code` (p. 1183).*
- class **OpenSSLServerSocketFactory**  
*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*
- class **OpenSSLSocket**  
*Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*
- class **OpenSSLSocketException**  
*Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.*
- class **OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*
- class **OpenSSLSocketInputStream**  
*An output stream for reading data from an OpenSSL Socket instance.*
- class **OpenSSLSocketOutputStream**  
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2858) instance.*

## 5.37 decaf::internal::net::tcp Namespace Reference

### Data Structures

- class **TcpSocket**  
*Platform-independent implementation of the socket interface.*
- class **TcpSocketInputStream**  
*Input stream for performing reads on a socket.*
- class **TcpSocketOutputStream**  
*Output stream for performing write operations on a socket.*

## 5.38 decaf::internal::nio Namespace Reference

### Data Structures

- class **BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 163) package to create the various default version of the NIO interfaces.*

- class **ByteBuffer**

*This class defines six categories of operations upon byte buffers:.*

- class **CharArrayBuffer**
- class **DoubleArrayBuffer**
- class **FloatArrayBuffer**
- class **IntArrayBuffer**
- class **LongArrayBuffer**
- class **ShortArrayBuffer**

## 5.39 decaf::internal::security Namespace Reference

### Data Structures

- class **SecureRandomImpl**

*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

## 5.40 decaf::internal::util Namespace Reference

### Namespaces

- namespace **concurrent**

### Data Structures

- class **ByteArrayAdapter**

*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*

- class **GenericResource**

*A Generic **Resource** (p. 3280) wraps some type and will delete it when the **Resource** (p. 3280) itself is deleted.*

- class **HexStringParser**

- class **Resource**

*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

- class **ResourceLifecycleManager**

- class **TimerTaskHeap**

*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

## 5.41 decaf::internal::util::concurrent Namespace Reference

### Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

- class **Transferer**

*Shared **internal** (p. 144) API for dual stacks and queues.*

- class **TransferQueue**

*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

- class **TransferStack**

## 5.42 decaf::io Namespace Reference

### Data Structures

- class **BlockingByteArrayInputStream**  
*This is a blocking version of a byte buffer stream.*
- class **BufferedInputStream**  
*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of **io** (p. 154) operations on the input stream.*
- class **BufferedOutputStream**  
*Wrapper around another output stream that buffers output before writing to the target output stream.*
- class **ByteArrayInputStream**  
*A **ByteArrayInputStream** (p. 1020) contains an **internal** (p. 144) buffer that contains bytes that may be read from the stream.*
- class **ByteArrayOutputStream**
- class **Closeable**  
*Interface for a class that implements the close method.*
- class **DataInput**  
*The **DataInput** (p. 1558) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*
- class **DataInputStream**  
*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*
- class **DataOutput**  
*The **DataOutput** (p. 1574) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*
- class **DataOutputStream**  
*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*
- class **EOFException**
- class **FileDescriptor**  
*This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*
- class **FilterInputStream**  
*A **FilterInputStream** (p. 1894) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*
- class **FilterOutputStream**  
*This class is the superclass of all classes that filter output streams.*



- class **Flushable**

*A **Flushable** (p. 1936) is a destination of data that can be flushed.*

- class **InputStream**

*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*

- class **InputStreamReader**

*An **InputStreamReader** (p. 2053) is a bridge from byte streams to character streams.*

- class **InterruptedIOException**

- class **IOException**

- class **OutputStream**

*Base interface for any class that wants to represent an output stream of bytes.*

- class **OutputStreamWriter**

*A class for turning a character stream into a byte stream.*

- class **PushbackInputStream**

*A **PushbackInputStream** (p. 3141) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

- class **Reader**

- class **UnsupportedEncodingException**

*Thrown when the the Character Encoding is not supported.*

- class **UTFDataFormatException**

*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*

- class **Writer**

## 5.43 decaf::lang Namespace Reference

### Namespaces

- namespace **exceptions**

### Data Structures

- class **Appendable**

*An object to which char sequences and values can be appended.*

- class **ArrayPointer**

*Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a **Type** and is **Thread** (p. 3765) Safe if the default Reference Counter is used.*

- class **ArrayPointerComparator**

*This implementation of **Comparator** is designed to allows objects in a **Collection** to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 730).*

- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

*A **CharSequence** (p. 1135) is a readable sequence of char values.*

- class **Comparable**

*This interface imposes a total ordering on the objects of each class that implements it.*

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2152) type for generic collections API calls.*

- class **Long**
- class **Math**

*The class **Math** (p. 2497) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2835) is the superclass of classes **Byte** (p. 960), **Double** (p. 1788), **Float** (p. 1904), **Integer** (p. 2077), **Long** (p. 2420), and **Short** (p. 3440).*

- struct **STATIC\_CAST\_TOKEN**
- struct **DYNAMIC\_CAST\_TOKEN**
- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a **Type** and is **Thread** (p. 3765) Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2946) instance.*

- class **Readable**

*A **Readable** (p. 3160) is a source of characters.*

- class **Runnable**

*Interface for a runnable object - defines a task that can be run by a thread.*

- class **Runtime**

- class **Short**

- class **String**

*The **String** (p. 3665) class represents an immutable sequence of chars.*

- class **System**

*The **System** (p. 3726) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

- class **Thread**

*A **Thread** (p. 3765) is a concurrent unit of execution.*

- class **ThreadGroup**

- class **Throwable**

*This class represents an error that has occurred.*

## Functions

- `template<typename T , typename R , typename U >`  
`bool operator== (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const U *left, const Pointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const U *left, const Pointer< T, R > &right)`

### 5.43.1 Function Documentation

**5.43.1.1** `template<typename T , typename R , typename U > bool  
decaf::lang::operator!= (const U * left, const Pointer< T, R > & right)  
[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.2** `template<typename T , typename R , typename U > bool  
decaf::lang::operator!= (const Pointer< T, R > & left, const U * right)  
[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.3** `template<typename T , typename R , typename U > bool  
decaf::lang::operator!= (const U * left, const ArrayPointer< T, R > & right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

**5.43.1.4** `template<typename T , typename R , typename U > bool  
decaf::lang::operator!= (const ArrayPointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

**5.43.1.5** `template<typename T , typename R , typename U > bool  
decaf::lang::operator== (const U * left, const Pointer< T, R > & right)  
[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.6** `template<typename T , typename R , typename U > bool  
decaf::lang::operator== (const Pointer< T, R > & left, const U * right)  
[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.7** `template<typename T , typename R , typename U > bool  
decaf::lang::operator== (const U * left, const ArrayPointer< T, R > & right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

**5.43.1.8** `template<typename T , typename R , typename U > bool  
decaf::lang::operator== (const ArrayPointer< T, R > & left, const U * right) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

## 5.44 decaf::lang::exceptions Namespace Reference

### Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NoSuchElementException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

## 5.45 decaf::net Namespace Reference

### Namespaces

- namespace **ssl**

### Data Structures

- class **BindException**
- class **ConnectException**
- class **HttpRetryException**
- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**

*Represents an IP address.*

- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

*This class implements server sockets.*

- class **ServerSocketFactory**

*Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*

- class **Socket**
- class **SocketAddress**

*Base class for protocol specific **Socket** (p. 3503) addresses.*

- class **SocketError**

*Static utility class to simplify handling of error codes for socket operations.*

- class **SocketException**

*Exception for errors when manipulating sockets.*

- class **SocketFactory**

*The **SocketFactory** (p. 3524) is used to create **Socket** (p. 3503) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

- class **SocketImpl**

*Acts as a base class for all physical **Socket** (p. 3503) implementations.*

- class **SocketImplFactory**

*Factory class interface for a Factory that creates SocketImpl objects.*

- class **SocketOptions**
- class **SocketTimeoutException**

- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p. 3921) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p. 3960) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

## 5.46 decaf::net::ssl Namespace Reference

### Data Structures

- class **SSLContext**  
*Represents an implementation of the Secure **Socket** (p. 3503) Layer for streaming based sockets.*
- class **SSLContextSpi**  
*Defines the interface that should be provided by an **SSLContext** (p. 3545) provider.*
- class **SSLParameters**
- class **SSLServerSocket**  
*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*
- class **SSLServerSocketFactory**  
*Factory class interface that provides methods to create SSL Server Sockets.*
- class **SSLSocket**
- class **SSLSocketFactory**  
*Factory class interface for a **SocketFactory** (p. 3524) that can create **SSLSocket** (p. 3563) objects.*



## 5.47 decaf::nio Namespace Reference

### Data Structures

- class **Buffer**

*A container for data of a specific primitive type.*

- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**

*This class defines six categories of operations upon byte buffers:.*

- class **CharBuffer**

*This class defines four categories of operations upon character buffers:.*

- class **DoubleBuffer**

*This class defines four categories of operations upon double buffers:.*

- class **FloatBuffer**

*This class defines four categories of operations upon float buffers:.*

- class **IntBuffer**

*This class defines four categories of operations upon int buffers:.*

- class **InvalidMarkException**

- class **LongBuffer**

*This class defines four categories of operations upon long long buffers:.*

- class **ReadOnlyBufferException**

- class **ShortBuffer**

*This class defines four categories of operations upon short buffers:.*

## 5.48 decaf::security Namespace Reference

### Namespaces

- namespace **auth**
- namespace **cert**

### Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 2293) interface is the top-level interface for all keys.*

- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

*Base interface for a principal, which can represent an individual or organization.*

- class **PublicKey**

*A public key.*

- class **SecureRandom**
- class **SecureRandomSpi**

*Interface class used by Security Service Providers to implement a source of secure random bytes.*

- class **SignatureException**

## 5.49 decaf::security::auth Namespace Reference

### Namespaces

- namespace **x500**

## 5.50 decaf::security::auth::x500 Namespace Reference

### Data Structures

- class **X500Principal**

## 5.51 decaf::security::cert Namespace Reference

### Data Structures

- class **Certificate**  
*Base interface for all identity certificates.*
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**  
*Base interface for all identity certificates.*

## 5.52 decaf::util Namespace Reference

### Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**
- namespace **zip**

### Data Structures

- class **AbstractCollection**  
*This class provides a skeletal implementation of the **Collection** (p. 1184) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**  
*This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **AbstractMap**  
*This class provides a skeletal implementation of the **Map** (p. 2459) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**  
*This class provides skeletal implementations of some **Queue** (p. 3149) operations.*
- class **AbstractSequentialList**  
*This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*
- class **AbstractSet**  
*This class provides a skeletal implementation of the **Set** (p. 3439) interface to minimize the effort required to implement this interface.*
- class **Collection**  
*The root interface in the collection hierarchy.*
- class **Comparator**  
*A comparison function, which imposes a total ordering on some collection of objects.*
- class **Date**  
*Wrapper class around a time value in milliseconds.*
- class **Iterator**  
*Defines an object that can be used to iterate over the elements of a collection.*
- class **List**  
*An ordered collection (also known as a sequence).*

- class **ListIterator**

*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*

- class **Map**

***Map** (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **PriorityQueue**

*An unbounded priority queue based on a binary heap algorithm.*

- class **Properties**

*Java-like properties class for mapping string names to string values.*

- class **Queue**

*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*

- class **Random**

***Random** (p. 3155) Value Generator which is used to generate a stream of pseudorandom numbers.*

- class **Set**

*A collection that contains no duplicate elements.*

- class **StlList**

***List** (p. 2337) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **StlMap**

***Map** (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **StlQueue**

*The **Queue** (p. 3149) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*

- class **StlSet**

***Set** (p. 3439) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*

- class **StringTokenizer**

- class **Timer**

*A facility for threads to schedule tasks for future execution in a background thread.*

- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3790).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3969)).*

## 5.53 decaf::util::comparators Namespace Reference

### Data Structures

- class **Less**

*Simple **Less** (p. 2328) **Comparator** (p. 1217) that compares to elements to determine if the first is less than the second.*



## 5.54 `decaf::util::concurrent` Namespace Reference

### Namespaces

- namespace **atomic**
- namespace **locks**

### Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BlockingQueue**

A `decaf::util::Queue` (p. 3149) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

- class **BrokenBarrierException**
- class **Callable**

A task that returns a result and may throw an exception.

- class **CancellationException**
- class **ConcurrentMap**

Interface for a **Map** (p. 2459) type that provides additional **atomic** (p. 173) `putIfAbsent`, `remove`, and `replace` methods alongside the already available **Map** (p. 2459) interface.

- class **ConcurrentStlMap**

**Map** (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

- class **CountDownLatch**
- class **Delayed**

A *mix-in* style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**
- class **Executor**

An object that executes submitted `decaf.lang Runnable` (p. 3325) tasks.

- class **ExecutorService**

An **Executor** (p. 1869) that provides methods to manage termination and methods that can produce a **Future** (p. 1966) for tracking progress of one or more asynchronous tasks.

- class **Future**

A **Future** (p. 1966) represents the result of an asynchronous computation.

- class **Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

**Mutex** (p. 2782) object that offers recursive support on all platforms as well as providing the ability to use the standard `wait / notify` pattern used in languages like Java.

- class **PooledThread**
- class **PooledThreadListener**  
*Abstract Listener Interface for users of **ThreadPool** (p. 3777).*
- class **RejectedExecutionException**
- class **RejectedExecutionHandler**  
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*
- class **Semaphore**  
*A counting semaphore.*
- class **Synchronizable**  
*The interface for all synchronizable objects (that is, objects that can be locked and unlocked).*
- class **SynchronousQueue**  
*A **blocking queue** (p. 843) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*
- class **TaskListener**
- class **ThreadFactory**  
*public interface **ThreadFactory** (p. 3774)*
- class **ThreadPool**  
*Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.*
- class **TimeoutException**
- class **TimeUnit**  
*A **TimeUnit** (p. 3807) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

## 5.55 decaf::util::concurrent::atomic Namespace Reference

### Data Structures

- class **AtomicBoolean**  
*A boolean value that may be updated atomically.*
- class **AtomicInteger**  
*An int value that may be updated atomically.*
- class **AtomicRefCounter**
- class **AtomicReference**  
*An Pointer reference that may be updated atomically.*

## 5.56 decaf::util::concurrent::locks Namespace Reference

### Data Structures

- class **Condition**

***Condition** (p. 1249) factors out the **Mutex** (p. 2782) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2377) implementations.*

- class **Lock**

***Lock** (p. 2377) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **LockSupport**

*Basic thread blocking primitives for creating **locks** (p. 174) and other synchronization classes.*

- class **ReadWriteLock**

*A **ReadWriteLock** (p. 3172) maintains a pair of associated **locks** (p. 174), one for read-only operations and one for writing.*

- class **ReentrantLock**

*A reentrant mutual exclusion **Lock** (p. 2377) with extended capabilities.*

## 5.57 decaf::util::logging Namespace Reference

### Data Structures

- class **ConsoleHandler**

*This **Handler** (p. 1978) publishes log records to `System.err`.*

- class **ErrorManager**

***ErrorManager** (p. 1828) objects can be attached to **Handlers** to process any error that occur on a **Handler** (p. 1978) during Logging.*

- class **Filter**

*A **Filter** (p. 1893) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

- class **Formatter**

*A **Formatter** (p. 1964) provides support for formatting **LogRecords**.*

- class **Handler**

*A **Handler** (p. 1978) object takes log messages from a **Logger** (p. 2386) and exports them.*

- class **Level**

*The **Level** (p. 2332) class defines a set of standard **logging** (p. 175) levels that can be used to control **logging** (p. 175) output.*

- class **Logger**

*A **Logger** (p. 2386) object is used to log messages for a specific system or application component.*

- class **LoggerHierarchy**

- class **LogManager**

*There is a single global **LogManager** (p. 2406) object that is used to maintain a set of shared state about **Loggers** and log services.*

- class **LogRecord**

***LogRecord** (p. 2413) objects are used to pass **logging** (p. 175) requests between the **logging** (p. 175) framework and individual log **Handlers**.*

- class **LogWriter**

- class **MarkBlockLogger**

*Defines a class that can be used to mark the entry and exit from scoped blocks.*

- class **PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 3126).*

- class **SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2413) in a human readable format.*

- class **SimpleLogger**

- class **StreamHandler**

*Stream based **logging** (p. 175) **Handler** (p. 1978).*

- class **XMLFormatter**

*Format a **LogRecord** (p. 2413) into a standard XML format.*

## Enumerations

- enum **Levels** {  
    **Off**, **Null**, **Markblock**, **Debug**,  
    **Info**, **Warn**, **Error**, **Fatal**,  
    **Throwing** }

*Defines an enumeration for logging levels.*

### 5.57.1 Enumeration Type Documentation

#### 5.57.1.1 enum decaf::util::logging::Levels

Defines an enumeration for **logging** (p. 175) levels.

##### Enumerator:

*Off*

*Null*

*Markblock*

*Debug*

*Info*

*Warn*

*Error*

*Fatal*

*Throwing*

## 5.58 decaf::util::zip Namespace Reference

### Data Structures

- class **Adler32**  
*Class that can be used to compute an Adler-32 **Checksum** (p. 1142) for a data stream.*
- class **CheckedInputStream**  
*An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1142) of the bytes read, the **Checksum** (p. 1142) can then be used to verify the integrity of the input stream.*
- class **CheckedOutputStream**  
*An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1142) of the bytes written, the **Checksum** (p. 1142) can then be used to verify the integrity of the output stream.*
- class **Checksum**  
*An interface used to represent **Checksum** (p. 1142) values in the Zip package.*
- class **CRC32**  
*Class that can be used to compute a CRC-32 checksum for a data stream.*
- class **DataFormatException**
- class **Deflater**  
*This class compresses data using the DEFLATE algorithm (see **specification**).*
- class **DeflaterOutputStream**  
*Provides a **FilterOutputStream** instance that compresses the data before writing it to the wrapped **OutputStream**.*
- class **Inflater**  
*This class uncompresses data that was compressed using the DEFLATE algorithm (see **specification**).*
- class **InflaterInputStream**  
*A **FilterInputStream** that decompresses data read from the wrapped **InputStream** instance.*
- class **ZipException**

## 5.59 std Namespace Reference

### Data Structures

- struct `less< decaf::lang::ArrayPointer< T > >`

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

- struct `less< decaf::lang::Pointer< T > >`

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*



## Chapter 6

# Data Structure Documentation

### 6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p. 1184) interface, to minimize the effort required to implement this interface.

#include <src/main/decaf/util/AbstractCollection.h> Inheritance diagram for decaf::util::AbstractCollection< E >:

#### Public Member Functions

- **AbstractCollection** ()
- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)  
*Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.*
- virtual bool **add** (const E &value DECAF\_UNUSED)  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Ensures that this collection contains the specified element (optional operation).*
- virtual bool **addAll** (const **Collection**< E > &collection)  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection (optional operation).*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).*
- virtual void **copy** (const **Collection**< E > &collection)  
*Renders this **Collection** (p. 1184) as a Copy of the given **Collection** (p. 1184).*

- virtual bool **contains** (const E &value) const throw ( lang::Exception )  
*Returns true if this collection contains the specified element.*
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **equals** (const **Collection**< E > &collection) const  
*Answers true if this **Collection** (p. 1184) and the one given are the same size and if each element contained in the **Collection** (p. 1184) given is equal to an element contained in this collection.*
- virtual bool **isEmpty** () const  
*Returns true if this collection contains no elements.*
- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from this collection, if it is present (optional operation).*
- virtual bool **removeAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes all of this collection's elements that are also contained in the specified collection (optional operation).*
- virtual bool **retainAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Retains only the elements in this collection that are contained in the specified collection (optional operation).*
- virtual std::vector< E > **toArray** () const  
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1184).*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Attributes

- util::concurrent::Mutex mutex

### 6.1.1 Detailed Description

**template<typename E> class decaf::util::AbstractCollection< E >**

This class provides a skeletal implementation of the **Collection** (p. 1184) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 1184) constructor, as per the recommendation in the **Collection** (p. 1184) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

**Since:**

1.0

## 6.1.2 Constructor & Destructor Documentation

**6.1.2.1** `template<typename E> decaf::util::AbstractCollection< E  
>::AbstractCollection () [inline]`

**6.1.2.2** `template<typename E> virtual decaf::util::AbstractCollection< E  
>::~~AbstractCollection () [inline, virtual]`

## 6.1.3 Member Function Documentation

**6.1.3.1** `template<typename E> virtual bool decaf::util::AbstractCollection<  
E >::add (const E &value DECAF_UNUSED) throw  
( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException,  
lang::exceptions::IllegalStateException ) [inline,  
virtual]`

Ensures that this collection contains the specified element (optional operation). Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 1184) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an `UnsupportedOperationException`.

### Parameters:

*value* - The element that must be ensured to be in this collection.

### Returns:

true if the collection was changed as a result of this call.

### Exceptions:

***UnsupportedOperationException*** if the add operation is not supported by this collection

***IllegalArgumentException*** if some property of the element prevents it from being added to this collection

***IllegalStateException*** if the element cannot be added at this time due to insertion restrictions

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::addAll()`,  
`decaf::util::AbstractCollection< cms::Connection * >::copy()`, and  
`decaf::util::AbstractCollection< cms::Connection * >::operator=()`.

```

6.1.3.2  template<typename E> virtual bool decaf::util::AbstractCollection<
        E >::addAll (const Collection< E > & collection) throw
        ( lang::exceptions::UnsupportedOperationException,
        lang::exceptions::IllegalArgumentException,
        lang::exceptions::IllegalStateException ) [inline,
        virtual]

```

Adds all of the elements in the specified collection to this collection (optional operation). The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

#### Parameters:

*collection* - The **Collection** (p.1184) whose elements are to be added to this **Collection** (p.1184).

#### Returns:

true if the collection was changed as a result of this call.

#### Exceptions:

*UnsupportedOperationException* if the `addAll` operation is not supported by this collection

*IllegalArgumentException* if some property of the element prevents it from being added to this collection

*IllegalStateException* if the element cannot be added at this time due to insertion restrictions

Reimplemented in `decaf::util::AbstractQueue< E >` (p.195).

```

6.1.3.3  template<typename E> virtual void decaf::util::AbstractCollection< E
        >::clear () throw ( lang::exceptions::UnsupportedOperationException )
        [inline, virtual]

```

Removes all of the elements from this collection (optional operation). The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.2155) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection is non-empty.

#### Exceptions:

*UnsupportedOperationException* if the `clear` operation is not supported by this collection

Implements **decaf::util::Collection**< **E** > (p. 1187).

Reimplemented in **decaf::util::AbstractQueue**< **E** > (p. 196), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3718), **decaf::util::PriorityQueue**< **E** > (p. 3031), **decaf::util::StlList**< **E** > (p. 3596), **decaf::util::StlSet**< **E** > (p. 3626), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3596), **decaf::util::StlList**< **CompositeTask** \* > (p. 3596), **decaf::util::StlList**< **URI** > (p. 3596), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3596), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3596), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3596), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3596), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3596), **decaf::util::StlList**< **cms::Destination** \* > (p. 3596), **decaf::util::StlList**< **cms::Session** \* > (p. 3596), **decaf::util::StlList**< **cms::Connection** \* > (p. 3596), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3626), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3626), **decaf::util::StlSet**< **Resource** \* > (p. 3626), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3626).

Referenced by **decaf::util::AbstractCollection**< **cms::Connection** \* >::copy(), and **decaf::util::AbstractCollection**< **cms::Connection** \* >::operator=().

**6.1.3.4** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & value) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element. This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

#### Parameters:

*value* - the value whose presence is to be queried for in this **Collection** (p. 1184).

#### Returns:

true if the value is contained in this collection

#### Exceptions:

*Exception* if an error occurs,

Implements **decaf::util::Collection**< **E** > (p. 1188).

Reimplemented in **decaf::util::StlList**< **E** > (p. 3596), **decaf::util::StlSet**< **E** > (p. 3627), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3596), **decaf::util::StlList**< **CompositeTask** \* > (p. 3596), **decaf::util::StlList**< **URI** > (p. 3596), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3596), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3596), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3596), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3596), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3596), **decaf::util::StlList**< **cms::Destination** \* > (p. 3596), **decaf::util::StlList**< **cms::Session** \* > (p. 3596), **decaf::util::StlList**< **cms::Connection** \* > (p. 3596), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3627), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3627), **decaf::util::StlSet**< **Resource** \* > (p. 3627), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3627).

Referenced by **decaf::util::AbstractCollection**< **cms::Connection** \* >::containsAll().

**6.1.3.5** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & collection) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection. This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

**Parameters:**

*collection* collection to be checked for containment in this collection

**Returns:**

true if this collection contains all of the elements in the specified collection.

**Exceptions:**

*Exception* if an error occurs,

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3719).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::equals()`.

**6.1.3.6** `template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & collection) [inline, virtual]`

Renders this **Collection** (p. 1184) as a Copy of the given **Collection** (p. 1184). This implementation iterates over the contents of the given collection adding each to this collection after first calling this Collection's clear method.

**Parameters:**

*collection* - the collection to mirror.

**6.1.3.7** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 1184) and the one given are the same size and if each element contained in the **Collection** (p. 1184) given is equal to an element contained in this collection.

**Parameters:**

*collection* - The **Collection** (p. 1184) to be compared to this one.

**Returns:**

true if this **Collection** (p. 1184) is equal to the one given.

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3720).

### 6.1.3.8 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements. This implementation returns `size()` (p. 1192) == 0.

#### Returns:

true if the size method return 0.

Implements `decaf::util::Collection< E >` (p. 1189).

Reimplemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3721), `decaf::util::StlList< E >` (p. 3598), `decaf::util::StlSet< E >` (p. 3627), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), `decaf::util::StlList< cms::Connection * >` (p. 3598), `decaf::util::StlSet< transport::TransportListener * >` (p. 3627), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3627), `decaf::util::StlSet< Resource * >` (p. 3627), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3627).

Referenced by `decaf::util::AbstractQueue< E >::clear()`.

### 6.1.3.9 `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

#### Exceptions:

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3701).

### 6.1.3.10 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3702).



**6.1.3.11** `template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3703).

**6.1.3.12** `template<typename E> AbstractCollection<E>& decaf::util::AbstractCollection< E >::operator= (const AbstractCollection< E > & collection) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

**Parameters:**

*collection* - the collection to copy

**Returns:**

a reference to this collection

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 3033).

**6.1.3.13** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::remove (const E & value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters:**

*value* - element to be removed from this collection, if present

**Returns:**

true if an element was removed as a result of this call

**Exceptions:**

***UnsupportedOperationException*** if the remove operation is not supported by this collection.

***IllegalArgumentException*** If the value is not a valid entry for this **Collection** (p. 1184).

Implements **decaf::util::Collection< E >** (p. 1190).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 3034), **decaf::util::StlList< E >** (p. 3600), **decaf::util::StlSet< E >** (p. 3628), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3600), **decaf::util::StlList< CompositeTask \* >** (p. 3600), **decaf::util::StlList< URI >** (p. 3600), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3600), **decaf::util::StlList< PrimitiveValueNode >** (p. 3600), **decaf::util::StlList< Pointer< Command > >** (p. 3600), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3600), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3600), **decaf::util::StlList< cms::Destination \* >** (p. 3600), **decaf::util::StlList< cms::Session \* >** (p. 3600), **decaf::util::StlList< cms::Connection \* >** (p. 3600), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3628), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3628), **decaf::util::StlSet< Resource \* >** (p. 3628), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3628).

```
6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::removeAll (const Collection< E > & collection)
           throw ( lang::exceptions::UnsupportedOperationException,
                   lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes all of this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

**Parameters:**

***collection*** - collection containing elements to be removed from this collection

**Returns:**

true if this collection changed as a result of the call

**Exceptions:**

***UnsupportedOperationException*** if the remove operation is not supported by this collection

***IllegalArgumentException***.

Reimplemented in **decaf::util::AbstractSet< E >** (p. 199), **decaf::util::AbstractSet< transport::TransportListener \* >** (p. 199), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 199), **decaf::util::AbstractSet< Resource \* >** (p. 199), and **decaf::util::AbstractSet< ActiveMQSession \* >** (p. 199).

```

6.1.3.15  template<typename E> virtual bool decaf::util::AbstractCollection<
            E >::retainAll (const Collection< E > & collection)
            throw ( lang::exceptions::UnsupportedOperationException,
                    lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

**Parameters:**

*collection* - collection containing elements to be retained in this collection

**Returns:**

true if this collection changed as a result of the call

**Exceptions:**

*UnsupportedOperationException* if the remove operation is not supported by this collection

*IllegalArgumentException*.

```

6.1.3.16  template<typename E> virtual std::vector<E>
            decaf::util::AbstractCollection< E >::toArray () const [inline, virtual]

```

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1184). All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

**Returns:**

an vector of copies of all the elements from this **Collection** (p. 1184)

Implements **decaf::util::Collection< E >** (p. 1192).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3724).

```

6.1.3.17  template<typename E> virtual bool decaf::util::AbstractCollection<
            E >::tryLock () throw ( decaf::lang::exceptions::RuntimeException )
            [inline, virtual]

```

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

```
6.1.3.18  template<typename E> virtual void decaf::util::AbstractCollection< E
          >::unlock () throw ( decaf::lang::exceptions::RuntimeException ) [inline,
          virtual]
```

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).

```
6.1.3.19  template<typename E> virtual void decaf::util::AbstractCollection<
          E >::wait (long long millisecs, int nanos) throw
          ( decaf::lang::exceptions::RuntimeException, de-
          caf::lang::exceptions::IllegalArgumentException,
          decaf::lang::exceptions::IllegalMonitorStateException,
          decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

**Exceptions:**

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.1.3.20** `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3708).

**6.1.3.21** `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3709).

## 6.1.4 Field Documentation

**6.1.4.1** `template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex [mutable, protected]`

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::lock()`, `decaf::util::AbstractCollection< cms::Connection * >::notify()`, `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`, `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`, `decaf::util::AbstractCollection< cms::Connection * >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

## 6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

#include <src/main/decaf/util/AbstractList.h> Inheritance diagram for decaf::util::AbstractList< E >:

### Public Member Functions

- virtual ~**AbstractList** ()

#### 6.2.1 Detailed Description

**template<typename E> class decaf::util::AbstractList< E >**

This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p. 198) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p. 1192) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1184) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

**Since:**

1.0

#### 6.2.2 Constructor & Destructor Documentation

**6.2.2.1 template<typename E > virtual decaf::util::AbstractList< E >::~~AbstractList () [inline, virtual]**

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractList.h**

## 6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 2459) interface, to minimize the effort required to implement this interface.

#include <src/main/decaf/util/AbstractMap.h> Inheritance diagram for decaf::util::AbstractMap< K, V, COMPARATOR >:

### Public Member Functions

- virtual ~AbstractMap ()

#### 6.3.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::AbstractMap< K, V, COMPARATOR >

This class provides a skeletal implementation of the **Map** (p. 2459) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the entrySet method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 199). This set should not support the add or remove methods, and its iterator should not support the remove method.

To implement a modifiable map, the programmer must additionally override this class's put method (which otherwise throws an UnsupportedOperationException), and the iterator returned by entrySet().iterator() must additionally implement its remove method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 2459) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since:

1.0

#### 6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<typename K , typename V , typename COMPARATOR > virtual decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/AbstractMap.h

## 6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 3149) operations.

#include <src/main/decaf/util/AbstractQueue.h> Inheritance diagram for decaf::util::AbstractQueue< E >:

### Public Member Functions

- **AbstractQueue** ()
- virtual ~**AbstractQueue** ()
- virtual bool **add** (const E &value) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.*
- virtual bool **addAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all the elements of a collection to the queue.*
- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves and removes the head of this queue.*
- virtual E **element** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves, but does not remove, the head of this queue.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*

### 6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p. 3149) operations. Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 3149) implementation that extends this class must minimally define a method **Queue** (p. 3149). **offer**(E) which does not permit insertion of null elements, along with methods **Queue** (p. 3149). **peek**() (p. 3151), **Queue.poll**() (p. 3151), **Collection.size**() (p. 1192), and a **Collection.iterator**() (p. 2152) supporting **Iterator.remove**() (p. 2155). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 179).

Since:

1.0



## 6.4.2 Constructor & Destructor Documentation

**6.4.2.1** `template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue()` [inline]

**6.4.2.2** `template<typename E > virtual decaf::util::AbstractQueue< E >::~~AbstractQueue()` [inline, virtual]

## 6.4.3 Member Function Documentation

**6.4.3.1** `template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & value) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )` [inline, virtual]

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available. This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

### Parameters:

*value* - the element to offer to the **Queue** (p. 3149).

### Returns:

true if the add succeeds.

### Exceptions:

*IllegalArgumentException* if the element cannot be added.

Implements `decaf::util::Collection< E >` (p. 1186).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 3031).

References `decaf::util::Queue< E >::offer()`.

**6.4.3.2** `template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )` [inline, virtual]

Adds all the elements of a collection to the queue. If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

### Parameters:

*collection* - the collection to be added to the queue.

**Returns:**

true if the operation succeeds.

**Exceptions:**

***IllegalArgumentException*** If the collection to be added to the queue is the queue itself.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 183).

**6.4.3.3** `template<typename E > virtual void decaf::util::AbstractQueue< E >::clear  
() throw ( lang::exceptions::UnsupportedOperationException ) [inline,  
virtual]`

Removes all elements of the queue. This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 183).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3718), and **decaf::util::PriorityQueue< E >** (p. 3031).

References **decaf::util::AbstractCollection< E >::isEmpty()**, and **decaf::util::Queue< E >::poll()**.

**6.4.3.4** `template<typename E > virtual E decaf::util::AbstractQueue< E >::element  
() const throw ( decaf::lang::exceptions::NoSuchElementException )  
[inline, virtual]`

Retrieves, but does not remove, the head of this queue. This method differs from peek only in that it throws an exception if this queue is empty.

This implementation returns the result of peek unless the queue is empty.

**Returns:**

the element in the head of the queue.

**Exceptions:**

***NoSuchElementException*** if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 3150).

References **decaf::util::Queue< E >::peek()**.

Referenced by **decaf::util::concurrent::SynchronousQueue< E >::drainTo()**.

**6.4.3.5** `template<typename E > virtual E decaf::util::AbstractQueue< E >::remove  
() throw ( decaf::lang::exceptions::NoSuchElementException ) [inline,  
virtual]`

Retrieves and removes the head of this queue. This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

**Returns:**

a copy of the element in the head of the queue.

**Exceptions:**

*NoSuchElementException* if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 3151).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 3034).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractQueue.h`

## 6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p.2337) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

#include <src/main/decaf/util/AbstractSequentialList.h> Inheritance diagram for decaf::util::AbstractSequentialList< E >:

### Public Member Functions

- virtual ~AbstractSequentialList ()

#### 6.5.1 Detailed Description

template<typename E> class decaf::util::AbstractSequentialList< E >

This class provides a skeletal implementation of the **List** (p.2337) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p.192) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p.192) class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the listIterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's hasNext, next, hasPrevious, previous and index methods.

For a modifiable list the programmer should additionally implement the list iterator's set method. For a variable-size list the programmer should additionally implement the list iterator's remove and add methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p.1184) interface specification.

Since:

1.0

#### 6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<typename E > virtual decaf::util::AbstractSequentialList< E >::~AbstractSequentialList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/AbstractSequentialList.h

## 6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 3439) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h> Inheritance      diagram      for
decaf::util::AbstractSet< E >:
```

### Public Member Functions

- virtual **~AbstractSet** ()
- virtual bool **removeAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes from this set all of its elements that are contained in the specified collection (optional operation).*

#### 6.6.1 Detailed Description

```
template<typename E> class decaf::util::AbstractSet< E >
```

This class provides a skeletal implementation of the **Set** (p. 3439) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p.1184) by extending **AbstractCollection** (p.179), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 3439) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since:

1.0

#### 6.6.2 Constructor & Destructor Documentation

**6.6.2.1** template<typename E> virtual decaf::util::AbstractSet< E >::~AbstractSet () [inline, virtual]

#### 6.6.3 Member Function Documentation

**6.6.3.1** template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const **Collection**< E > & collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]

Removes from this set all of its elements that are contained in the specified collection (optional operation). If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates

over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

**Parameters:**

*collection* - The **Collection** (p. 1184) whose elements are to be retained

**Returns:**

true if the collection changed as a result of this call

**Exceptions:**

***UnsupportedOperationException***

***IllegalArgumentException***

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 188).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSet.h`

## 6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 3889) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3889) instances.

#include <src/main/activemq/transport/AbstractTransportFactory.h> Inheritance diagram for activemq::transport::AbstractTransportFactory:

### Public Member Functions

- virtual **~AbstractTransportFactory** ()

### Protected Member Functions

- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::NoSuchElementException** )

*Creates the WireFormat that is configured for this **Transport** (p. 3883) and returns it.*

#### 6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 3889) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3889) instances.

Since:

3.0

#### 6.7.2 Constructor & Destructor Documentation

- 6.7.2.1** virtual **activemq::transport::AbstractTransportFactory::~~AbstractTransportFactory** () [inline, virtual]

#### 6.7.3 Member Function Documentation

- 6.7.3.1** virtual **Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw ( **decaf::lang::exceptions::NoSuchElementException** ) [protected, virtual]

Creates the WireFormat that is configured for this **Transport** (p.3883) and returns it. The default WireFormat is Openwire.

Parameters:

*properties* The properties that were configured on the URI.

**Returns:**

a pointer to a `WireFormat` instance that the caller then owns.

**Exceptions:**

***NoSuchElementException*** if the configured `WireFormat` is not found.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/AbstractTransportFactory.h`



## 6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

### Public Member Functions

- virtual `~ActiveMQAckHandler ()`
- virtual void `acknowledgeMessage (const commands::Message *message)=0` throw ( cms::CMSException )

*Method called to acknowledge the message passed.*

### 6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

**Since:**

2.0

### 6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 virtual `activemq::core::ActiveMQAckHandler::~~ActiveMQAckHandler ()`  
[inline, virtual]

### 6.8.3 Member Function Documentation

- 6.8.3.1 virtual void `activemq::core::ActiveMQAckHandler::acknowledgeMessage (const commands::Message * message)` throw ( cms::CMSException )  
[pure virtual]

Method called to acknowledge the message passed.

**Parameters:**

*message* Message to Acknowledge

**Exceptions:**

*CMSException*

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQAckHandler.h`

## 6.9 activemq::commands::ActiveMQBlobMessage Class Reference

#include <src/main/activemq/commands/ActiveMQBlobMessage.h> Inheritance diagram for activemq::commands::ActiveMQBlobMessage:

### Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual **~ActiveMQBlobMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBlobMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- std::string **getRemoteBlobUrl** () const  
*Get the Remote URL of the Blob.*
- void **setRemoteBlobUrl** (const std::string &remoteURL)  
*Set the Remote URL of the Blob.*
- std::string **getMimeType** () const  
*Get the Mime Type of the Blob.*
- void **setMimeType** (const std::string &mimeType)  
*Set the Mime Type of the Blob.*
- std::string **getName** () const  
*Gets the Name of the Blob.*
- void **setName** (const std::string &name)  
*Sets the Name of the Blob.*

- bool **isDeletedByBroker** () const  
*Gets if this Blob is deleted by the Broker.*
- void **setDeletedByBroker** (bool value)  
*Sets the Deleted By Broker flag.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY\_MIME\_TYPE**

## 6.9.1 Constructor & Destructor Documentation

**6.9.1.1** `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()`

**6.9.1.2** `virtual  
activemq::commands::ActiveMQBlobMessage::~ActiveMQBlobMessage ()  
[inline, virtual]`

## 6.9.2 Member Function Documentation

**6.9.2.1** `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone  
() const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns:

new copy of this message

Implements **cms::Message** (p. 2539).

**6.9.2.2** `virtual ActiveMQBlobMessage* ac-  
tivismq::commands::ActiveMQBlobMessage::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2520).

**6.9.2.3** `virtual void ac-  
tivismq::commands::ActiveMQBlobMessage::copyDataStructure (const  
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2521).

**6.9.2.4** `virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 429).

**6.9.2.5** `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from `activemq::commands::Message` (p. 2523).

**6.9.2.6** `std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const` [inline]

Get the Mime Type of the Blob.

**Returns:**

string holding the MIME Type.

**6.9.2.7** `std::string activemq::commands::ActiveMQBlobMessage::getName () const` [inline]

Gets the Name of the Blob.

**Returns:**

string name of the Blob.

**6.9.2.8** `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const` [inline]

Get the Remote URL of the Blob.

**Returns:**

string from of the Remote Blob URL.

**6.9.2.9** `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const` [inline]

Gets if this Blob is deleted by the Broker.

**Returns:**

true if the Blob is deleted by the Broker.

**6.9.2.10** `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool value)` [inline]

Sets the Deleted By Broker flag.

**Parameters:**

*value* - set the Delete by broker flag to value.

**6.9.2.11** `void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & mimeType)` [inline]

Set the Mime Type of the Blob.

**Parameters:**

*mimeType* - String holding the MIME Type.

**6.9.2.12** `void activemq::commands::ActiveMQBlobMessage::setName (const std::string & name)` [inline]

Sets the Name of the Blob.

**Parameters:**

*name* - Name of the Blob.

**6.9.2.13** `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & remoteURL)` [inline]

Set the Remote URL of the Blob.

**Parameters:**

*remoteURL* - String form of the Remote URL.

**6.9.2.14** `virtual std::string activemq::commands::ActiveMQBlobMessage::toString  
() const [virtual]`

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2530).

### 6.9.3 Field Documentation

**6.9.3.1** `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_-  
MIME_TYPE [static]`

**6.9.3.2** `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_-  
ACTIVEMQBLOBMESSAGE = 29 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBlobMessage.h`

## 6.10 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.209).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.10.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.209). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.10.2 Constructor & Destructor Documentation

**6.10.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.10.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

## 6.10.3 Member Function Documentation

**6.10.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.10.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.10.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::marshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.10.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.10.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700).

**6.10.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

**6.10.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h`

## 6.11 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.213).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.11.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.213). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.11.2 Constructor & Destructor Documentation

**6.11.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.11.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

## 6.11.3 Member Function Documentation

**6.11.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.11.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.11.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::marshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.11.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.11.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2715).

**6.11.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

**6.11.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h`

## 6.12 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.217).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.12.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.217). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.12.2 Constructor & Destructor Documentation

**6.12.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.12.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

## 6.12.3 Member Function Documentation

**6.12.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.12.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.12.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.12.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.12.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2710).

**6.12.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p.2711).

**6.12.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p.2711).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h`

## 6.13 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.221).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.13.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.221). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.13.2 Constructor & Destructor Documentation

**6.13.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.13.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

### 6.13.3 Member Function Documentation

**6.13.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

#### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.13.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.13.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::marshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.13.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.13.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705).

**6.13.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

**6.13.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h`

## 6.14 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.225).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.14.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.225). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.14.2 Constructor & Destructor Documentation

**6.14.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.14.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

## 6.14.3 Member Function Documentation

**6.14.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.14.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.14.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.14.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.14.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2695).

**6.14.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

**6.14.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h`

## 6.15 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.229).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.15.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.229). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.15.2 Constructor & Destructor Documentation

**6.15.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

**6.15.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

## 6.15.3 Member Function Documentation

**6.15.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.15.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.15.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.15.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.15.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720).

**6.15.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

**6.15.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h`

## 6.16 activemq::commands::ActiveMQBytesMessage Class Reference

#include <src/main/activemq/commands/ActiveMQBytesMessage.h> Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

### Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBytesMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const throw ( cms::MessageNotReadableException, cms::CMSException )  
*Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.*
- virtual int **getBodyLength** () const throw ( cms::MessageNotReadableException, cms::CMSException )  
*Returns the number of bytes contained in the body of this message.*

- virtual void **reset** () throw ( cms::MessageFormatException, cms::CMSEException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Bytes message stream.*
- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the bytes message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Bytes message stream.*
- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the bytes message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the bytes message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the bytes message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the bytes message stream.*
- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Bytes message stream.*
- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a char to the bytes message stream as a 1-byte value.*
- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Bytes message stream.*



- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the bytes message stream as a 4 byte value.*
- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Bytes message stream.*
- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a double to the bytes message stream as a 8 byte value.*
- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit signed short from the Bytes message stream.*
- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed short to the bytes message stream as a 2 byte value.*
- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit unsigned short from the Bytes message stream.*
- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a unsigned short to the bytes message stream as a 2 byte value.*
- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit signed integer from the Bytes message stream.*
- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed int to the bytes message stream as a 4 byte value.*
- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Bytes message stream.*
- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a long long to the bytes message stream as a 8 byte value.*
- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an ASCII String from the Bytes message stream.*
- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an ASCII String to the Bytes message stream.*

- virtual std::string **readUTF** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an UTF String from the BytesMessage stream.*

- virtual void **writeUTF** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an UTF String to the BytesMessage stream.*

## Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQBYTESMESSAGE** = 24

### 6.16.1 Constructor & Destructor Documentation

**6.16.1.1** `activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()`

**6.16.1.2** `virtual  
activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage  
( ) [virtual]`

### 6.16.2 Member Function Documentation

**6.16.2.1** `virtual void activemq::commands::ActiveMQBytesMessage::clearBody ()  
throw ( cms::CMSEException ) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 428).

**6.16.2.2** `virtual cms::BytesMessage* ac-  
tivemq::commands::ActiveMQBytesMessage::clone () const  
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns:

new copy of this message

Implements `cms::BytesMessage` (p. 1059).

**6.16.2.3** `virtual ActiveMQBytesMessage* ac-  
tivemq::commands::ActiveMQBytesMessage::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2520).

**6.16.2.4 virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const DataStructure \* src) [virtual]**

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2521).

**6.16.2.5 virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure \* value) const [virtual]**

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Parameters:**

*value* The **Command** (p. 1194) to compare to this one.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 429).

**6.16.2.6 virtual unsigned char\* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const throw ( cms::MessageNotReadableException, cms::CMSEException ) [virtual]**

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer. Call **getBodyLength** to determine the number of bytes to expect.

**Returns:**

const pointer to a byte buffer

**Exceptions:**

*CMSEException* - If an internal error occurs.

*MessageNotReadableException* - If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 1059).

**6.16.2.7** `virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength  
() const throw ( cms::MessageNotReadableException, cms::CMSException  
) [virtual]`

Returns the number of bytes contained in the body of this message.

**Returns:**

number of bytes.

**Exceptions:**

*CMSException* - If an internal error occurs.

*MessageNotReadableException* - If the message is in Write Only Mode.

Implements `cms::BytesMessage` (p. 1060).

**6.16.2.8** `virtual unsigned char ac-  
tivemq::commands::ActiveMQBytesMessage::getDataStructureType ()  
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new `DataStructure` (p. 1660) type copy.

Reimplemented from `activemq::commands::Message` (p. 2523).

**6.16.2.9** `virtual void activemq::commands::ActiveMQBytesMessage::onSend ()  
[virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<  
cms::BytesMessage >` (p. 436).

**6.16.2.10** `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean  
() const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a Boolean from the Bytes message stream.

**Returns:**

boolean value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 1060).

**6.16.2.11** `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte() const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a Byte from the Bytes message stream.

**Returns:**

unsigned char value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::BytesMessage` (p.1060).

**6.16.2.12** `virtual int activemq::commands::ActiveMQBytesMessage::readBytes(unsigned char * buffer, int length) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

**Parameters:**

*buffer* the buffer into which the data is read

*length* the number of bytes to read; must be less than or equal to value.length

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::BytesMessage` (p.1061).

**6.16.2.13** `virtual int activemq::commands::ActiveMQBytesMessage::readBytes  
(std::vector< unsigned char > & value) const throw (  
cms::MessageEOFException, cms::MessageNotReadableException,  
cms::CMSEException )` [virtual]

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

**Parameters:**

*value* buffer to place data in

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::BytesMessage` (p.1061).

**6.16.2.14** `virtual char activemq::commands::ActiveMQBytesMessage::readChar  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException )` [virtual]

Reads a Char from the Bytes message stream.

**Returns:**

char value from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::BytesMessage` (p.1062).

**6.16.2.15** `virtual double ac-  
tivismq::commands::ActiveMQBytesMessage::readDouble  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException )` [virtual]

Reads a 64 bit double from the Bytes message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1062).

**6.16.2.16** virtual float activemq::commands::ActiveMQBytesMessage::readFloat  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]

Reads a 32 bit float from the Bytes message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1063).

**6.16.2.17** virtual int activemq::commands::ActiveMQBytesMessage::readInt  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]

Reads a 32 bit signed integer from the Bytes message stream.

**Returns:**

int value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1063).

**6.16.2.18** virtual long long ac-  
tivismq::commands::ActiveMQBytesMessage::readLong  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]

Reads a 64 bit long from the Bytes message stream.

**Returns:**

long long value from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1063).

**6.16.2.19** `virtual short activemq::commands::ActiveMQBytesMessage::readShort  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a 16 bit signed short from the Bytes message stream.

**Returns:**

short value from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1064).

**6.16.2.20** `virtual std::string ac-  
tivismq::commands::ActiveMQBytesMessage::readString  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads an ASCII String from the Bytes message stream.

**Returns:**

String from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::BytesMessage** (p.1064).



**6.16.2.21** virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

**Returns:**

unsigned short value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements cms::BytesMessage (p.1064).

**6.16.2.22** virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]

Reads an UTF String from the BytesMessage stream.

**Returns:**

String from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements cms::BytesMessage (p.1065).

**6.16.2.23** virtual void activemq::commands::ActiveMQBytesMessage::reset ()  
throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

**Exceptions:**

*CMSException* - If the provider fails to perform the reset operation.

*MessageFormatException* - If the Message (p.2516) has an invalid format.

Implements cms::BytesMessage (p.1065).

**6.16.2.24** `virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * buffer, int numBytes) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

sets the bytes given to the message body.

**Parameters:**

*buffer* Byte Buffer to copy

*numBytes* Number of bytes in Buffer to copy

**Exceptions:**

*CMSException* - If an internal error occurs.

*MessageNotWriteableException* - if in Read Only Mode.

Implements `cms::BytesMessage` (p.1065).

**6.16.2.25** `virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p.2530).

**6.16.2.26** `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters:**

*value* boolean to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::BytesMessage` (p.1066).

**6.16.2.27** `virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

**Parameters:**

*value* byte to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p.1066).

**6.16.2.28** virtual void activemq::commands::ActiveMQBytesMessage::writeBytes  
(const unsigned char \* *value*, int *offset*, int *length*) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

*offset* the initial offset within the byte array

*length* the number of bytes to use

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p.1066).

**6.16.2.29** virtual void activemq::commands::ActiveMQBytesMessage::writeBytes  
(const std::vector< unsigned char > & *value*) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p.1067).

**6.16.2.30** virtual void activemq::commands::ActiveMQBytesMessage::writeChar  
(char *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a char to the bytes message stream as a 1-byte value.

**Parameters:**

*value* char to write to the stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1067).

**6.16.2.31** virtual void activemq::commands::ActiveMQBytesMessage::writeDouble  
(double *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSEException ) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

**Parameters:**

*value* double to write to the stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1067).

**6.16.2.32** virtual void activemq::commands::ActiveMQBytesMessage::writeFloat  
(float *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSEException ) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

**Parameters:**

*value* float to write to the stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1068).

**6.16.2.33** virtual void activemq::commands::ActiveMQBytesMessage::writeInt (int  
*value*) throw ( cms::MessageNotWriteableException, cms::CMSEException  
) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

**Parameters:**

*value* signed int to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1068).

**6.16.2.34** virtual void activemq::commands::ActiveMQBytesMessage::writeLong  
(long long *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

**Parameters:**

*value* signed long long to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1068).

**6.16.2.35** virtual void activemq::commands::ActiveMQBytesMessage::writeShort  
(short *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

**Parameters:**

*value* signed short to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1069).

**6.16.2.36** virtual void activemq::commands::ActiveMQBytesMessage::writeString  
(const std::string & *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes an ASCII String to the Bytes message stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1069).

**6.16.2.37** virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

**Parameters:**

*value* unsigned short to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements cms::BytesMessage (p.1069).

**6.16.2.38** virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes an UTF String to the BytesMessage stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* - if the message is in write-only mode.

Implements cms::BytesMessage (p.1070).

## 6.16.3 Field Documentation

**6.16.3.1** const unsigned char activemq::commands::ActiveMQBytesMessage::ID\_ - ACTIVEMQBYTESMESSAGE = 24 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBytesMessage.h**

## 6.17

activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

Class Reference

6.17 ~~activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller~~<sup>249</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.249).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.17.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.249). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.17.2 Constructor & Destructor Documentation

**6.17.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.17.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.17.3 Member Function Documentation

**6.17.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.17.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.17.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.17

**activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

**Class Reference**

251

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2699).

### 6.17.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2699).

### 6.17.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2700).

**6.17.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

**6.17.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h`

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.253).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h> Include diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.18.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.253). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.18.2 Constructor & Destructor Documentation

**6.18.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.18.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.18.3 Member Function Documentation

**6.18.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.18.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.18.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.18

**activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

**Class Reference**

255

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2714).

### 6.18.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2714).

### 6.18.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2715).

**6.18.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

**6.18.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h`

## 6.19

activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

Class Reference

6.19 ~~activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller~~<sup>257</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.257).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h> Include diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.19.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.257). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.19.2 Constructor & Destructor Documentation

**6.19.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.19.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.19.3 Member Function Documentation

**6.19.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.19.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.19.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.19

**activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

**Class Reference**

259

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2709).

### 6.19.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2709).

### 6.19.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2710).

**6.19.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

**6.19.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h`

## 6.20

activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller

Class Reference

6.20 ~~activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller~~<sup>261</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.261).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h> Include diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.20.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.261). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.20.2 Constructor & Destructor Documentation

**6.20.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.20.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.20.3 Member Function Documentation

**6.20.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.20.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.20.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.20

**activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

**Class Reference**

**263**

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2694).

### 6.20.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2694).

### 6.20.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2695).

**6.20.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

**6.20.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h`

## 6.21

activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller

Class Reference

6.21 ~~activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller~~<sup>265</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.265).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h> Include diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.21.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.265). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.21.2 Constructor & Destructor Documentation

**6.21.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.21.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.21.3 Member Function Documentation

**6.21.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.21.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.21.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.21

**activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

**Class Reference**

**267**

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2719).

### 6.21.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2719).

### 6.21.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2720).

**6.21.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

**6.21.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h`

## 6.22

activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

Class Reference

6.22 ~~activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller~~<sup>269</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.269).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.22.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.269). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.22.2 Constructor & Destructor Documentation

**6.22.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

**6.22.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

## 6.22.3 Member Function Documentation

**6.22.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.22.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.22.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.22

**activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

**Class Reference**

**271**

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2704).

### 6.22.3.4 virtual void ac-

```
timemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2704).

### 6.22.3.5 virtual int ac-

```
timemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2705).

**6.22.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

**6.22.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h`

## 6.23 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

#include <src/main/activemq/core/ActiveMQConnection.h> Inheritance diagram for activemq::core::ActiveMQConnection:

### Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)  
*Constructor.*
- virtual ~**ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** \*session) throw ( cms::CMSException )  
*Removes the session resources for the given session instance.*
- virtual void **addProducer** (**ActiveMQProducer** \*producer) throw ( cms::CMSException )  
*Adds an active Producer to the Set of known producers.*
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw ( cms::CMSException )  
*Removes an active Producer to the Set of known producers.*
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** \*dispatcher) throw ( cms::CMSException )  
*Adds a dispatcher for a consumer.*
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw ( cms::CMSException )  
*Removes the dispatcher for a consumer.*
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** \*consumer, long long timeout) throw ( exceptions::ActiveMQException )  
*If supported sends a message pull request to the service provider asking for the delivery of a new message.*
- bool **isClosed** () const  
*Checks if this connection has been closed.*
- bool **isStarted** () const  
*Check if this connection has been started.*
- bool **isTransportFailed** () const  
*Checks if the Connection's Transport has failed.*

- virtual void **destroyDestination** (const **commands::ActiveMQDestination** \*destination) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalStateException**, **decaf::lang::exceptions::UnsupportedOperationException**, **activemq::exceptions::ActiveMQException** )

*Requests that the Broker removes the given Destination.*

- virtual void **destroyDestination** (const **cms::Destination** \*destination) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalStateException**, **decaf::lang::exceptions::UnsupportedOperationException**, **activemq::exceptions::ActiveMQException** )

*Requests that the Broker removes the given Destination.*

- virtual const **cms::ConnectionMetaData** \* **getMetaData** () const throw ( **cms::CMSEException** )

*Gets the metadata for this connection.*

- virtual **cms::Session** \* **createSession** () throw ( **cms::CMSEException** )

*Creates a new Session to work for this Connection.*

- virtual std::string **getClientID** () const

*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the **setClientID** method.*

**Returns:**

*Client Id String for this **Connection** (p. 1262).*

**Exceptions:**

**CMSEException** (p. 1160) if the provider fails to return the client id or an internal error occurs.

- virtual void **setClientID** (const std::string &clientID)

*Sets the client identifier for this connection.*

*The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1324) object and transparently assigned to the **Connection** (p. 1262) object it creates.*

*If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1997).*

**Parameters:**

**clientID** *The unique client identifier to assign to the **Connection** (p. 1262).*

**Exceptions:**

**CMSEException** (p. 1160) if the provider fails to set the client id due to some internal error.

**InvalidClientIDException** if the id given is somehow invalid or is a duplicate.

**IllegalStateException** (p. 1997) if the client tries to set the id after a **Connection** (p. 1262) method has been called.

- virtual **cms::Session** \* **createSession** (**cms::Session::AcknowledgeMode** ackMode) throw ( **cms::CMSEException** )

*Creates a new Session to work for this Connection using the specified acknowledgment mode.*

- virtual void **close** () throw ( **cms::CMSEException** )

*Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).*



- virtual void **start** () throw ( cms::CMSEException )  
*Starts or (restarts) a connections delivery of incoming messages.*
- virtual void **stop** () throw ( cms::CMSEException )  
*Stop the flow of incoming messages.*
- virtual cms::ExceptionListener \* **getExceptionListener** () const  
*Gets the registered Exception Listener for this connection.*
- virtual void **setExceptionListener** (cms::ExceptionListener \*listener)  
*Sets the registered Exception Listener for this connection.*
- void **setUsername** (const std::string &username)  
*Sets the username that should be used when creating a new connection.*
- const std::string & **getUsername** () const  
*Gets the username that this factory will use when creating a new connection instance.*
- void **setPassword** (const std::string &password)  
*Sets the password that should be used when creating a new connection.*
- const std::string & **getPassword** () const  
*Gets the password that this factory will use when creating a new connection instance.*
- void **setDefaultClientId** (const std::string &clientId)  
*Sets the Client Id.*
- void **setBrokerURL** (const std::string &brokerURL)  
*Sets the Broker URL that should be used when creating a new connection instance.*
- const std::string & **getBrokerURL** () const  
*Gets the Broker URL that this factory will use when creating a new connection instance.*
- void **setPrefetchPolicy** (PrefetchPolicy \*policy)  
*Sets the **PrefetchPolicy** (p. 2976) instance that this factory should use when it creates new Connection instances.*
- PrefetchPolicy \* **getPrefetchPolicy** () const  
*Gets the pointer to the current **PrefetchPolicy** (p. 2976) that is in use by this ConnectionFactory.*
- void **setRedeliveryPolicy** (RedeliveryPolicy \*policy)  
*Sets the **RedeliveryPolicy** (p. 3177) instance that this factory should use when it creates new Connection instances.*
- RedeliveryPolicy \* **getRedeliveryPolicy** () const  
*Gets the pointer to the current **RedeliveryPolicy** (p. 3177) that is in use by this ConnectionFactory.*

- bool **isDispatchAsync** () const
- void **setDispatchAsync** (bool value)  
*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*
- bool **isAlwaysSyncSend** () const  
*Gets if the Connection should always send things Synchronously.*
- void **setAlwaysSyncSend** (bool value)  
*Sets if the Connection should always send things Synchronously.*
- bool **isUseAsyncSend** () const  
*Gets if the useAsyncSend option is set.*
- void **setUseAsyncSend** (bool value)  
*Sets the useAsyncSend option.*
- bool **isUseCompression** () const  
*Gets if the Connection is configured for Message body compression.*
- void **setUseCompression** (bool value)  
*Sets whether Message body compression is enabled.*
- unsigned int **getSendTimeout** () const  
*Gets the assigned send timeout for this Connector.*
- void **setSendTimeout** (unsigned int timeout)  
*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*
- unsigned int **getCloseTimeout** () const  
*Gets the assigned close timeout for this Connector.*
- void **setCloseTimeout** (unsigned int timeout)  
*Sets the close timeout to use when sending the disconnect request.*
- unsigned int **getProducerWindowSize** () const  
*Gets the configured producer window size for Producers that are created from this connector.*
- void **setProducerWindowSize** (unsigned int windowSize)  
*Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.*
- long long **getNextSessionId** ()  
*Get the Next available Session Id.*
- long long **getNextTempDestinationId** ()  
*Get the Next Temporary Destination Id.*
- long long **getNextLocalTransactionId** ()

*Get the Next Temporary Destination Id.*

- void **addTransportListener** (**transport::TransportListener** \*transportListener)  
*Adds a **transport** (p. 97) listener so that a client can be notified of events in the underlying **transport** (p. 97), client's are always notified after the event has been processed by the **Connection** class.*
- void **removeTransportListener** (**transport::TransportListener** \*transportListener)  
*Removes a registered **TransportListener** from the **Connection**'s set of **Transport** listeners, this listener will no longer receive any **Transport** related events.*
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)  
*Event handler for the receipt of a non-response command from the **transport** (p. 97).*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **transportInterrupted** ()  
*The **transport** (p. 97) has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The **transport** (p. 97) has resumed after an interruption.*
- const **commands::ConnectionInfo** & **getConnectionInfo** () const throw ( exceptions::ActiveMQException )  
*Gets the **ConnectionInfo** for this **Object**, if the **Connection** is not open than this method throws an exception.*
- const **commands::ConnectionId** & **getConnectionId** () const throw ( exceptions::ActiveMQException )  
*Gets the **ConnectionId** for this **Object**, if the **Connection** is not open than this method throws an exception.*
- **transport::Transport** & **getTransport** () const  
*Gets a reference to this object's **Transport** instance.*
- void **oneway** (**Pointer**< **commands::Command** > command) throw ( activemq::exceptions::ActiveMQException )  
*Sends a oneway message.*
- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw ( activemq::exceptions::ActiveMQException )  
*Sends a synchronous request and returns the response from the broker.*
- virtual void **fire** (const exceptions::ActiveMQException &ex)  
*Notify the exception listener.*
- void **setTransportInterruptionProcessingComplete** ()  
*Indicates that a **Connection** resource that is processing the **transportInterrupted** event has completed.*

### 6.23.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

Since:

2.0

### 6.23.2 Constructor & Destructor Documentation

**6.23.2.1** `activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & transport, const Pointer< decaf::util::Properties > & properties)`

Constructor.

Parameters:

*transport* (p. 97) The Transport requested for this connection to the Broker.  
*properties* The Properties that were defined for this connection

**6.23.2.2** `virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection ()`  
 [virtual]

### 6.23.3 Member Function Documentation

**6.23.3.1** `virtual void activemq::core::ActiveMQConnection::addDispatcher (const Pointer< commands::ConsumerId > & consumer, Dispatcher * dispatcher) throw ( cms::CMSException )` [virtual]

Adds a dispatcher for a consumer.

Parameters:

*consumer* - The consumer for which to register a dispatcher.  
*dispatcher* - The dispatcher to handle incoming messages for the consumer.

**6.23.3.2** `virtual void activemq::core::ActiveMQConnection::addProducer (ActiveMQProducer * producer) throw ( cms::CMSException )` [virtual]

Adds an active Producer to the Set of known producers.

Parameters:

*producer* - The Producer to add from the the known set.

**6.23.3.3** `void activemq::core::ActiveMQConnection::addTransportListener (transport::TransportListener * transportListener)`

Adds a **transport** (p.97) listener so that a client can be notified of events in the underlying **transport** (p.97), client's are always notified after the event has been processed by the Connection class. Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

**Parameters:**

*transportListener* The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.

#### 6.23.3.4 virtual void activemq::core::ActiveMQConnection::close () throw ( cms::CMSException ) [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

**Exceptions:**

*CMSException*

Implements **cms::Connection** (p. 1263).

#### 6.23.3.5 virtual cms::Session\* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode *ackMode*) throw ( cms::CMSException ) [virtual]

Creates a new Session to work for this Connection using the specified acknowledgment mode.

**Parameters:**

*ackMode* the Acknowledgment Mode to use.

**Exceptions:**

*CMSException*

Implements **cms::Connection** (p. 1264).

#### 6.23.3.6 virtual cms::Session\* activemq::core::ActiveMQConnection::createSession () throw ( cms::CMSException ) [virtual]

Creates a new Session to work for this Connection.

**Exceptions:**

*CMSException*

Implements **cms::Connection** (p. 1264).

#### 6.23.3.7 virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination \* *destination*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException ) [virtual]

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

**Parameters:**

*destination* The CMS Destination the Broker will be requested to remove.

**Exceptions:**

*NullPointerException* If the passed Destination is Null

*IllegalStateException* If the connection is closed.

*UnsupportedOperationException* If the wire format in use does not support this operation.

*ActiveMQException* If any other error occurs during the attempt to destroy the destination.

```
6.23.3.8  virtual void activemq::core::ActiveMQConnection::destroyDestination
          (const commands::ActiveMQDestination * destination)
          throw ( decaf::lang::exceptions::NullPointerException,
                  decaf::lang::exceptions::IllegalStateException, de-
                  ccaf::lang::exceptions::UnsupportedOperationException,
                  activemq::exceptions::ActiveMQException ) [virtual]
```

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

**Parameters:**

*destination* The Destination the Broker will be requested to remove.

**Exceptions:**

*NullPointerException* If the passed Destination is Null

*IllegalStateException* If the connection is closed.

*UnsupportedOperationException* If the wire format in use does not support this operation.

*ActiveMQException* If any other error occurs during the attempt to destroy the destination.

```
6.23.3.9  virtual void activemq::core::ActiveMQConnection::fire (const
          exceptions::ActiveMQException & ex) [virtual]
```

Notify the exception listener.

**Parameters:**

*ex* the exception to fire

```
6.23.3.10  const std::string& activemq::core::ActiveMQConnection::getBrokerURL
          () const
```

Gets the Broker URL that this factory will use when creating a new connection instance.

**Returns:**

brokerURL string

### 6.23.3.11 `virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the `setClientID` method.

#### Returns:

Client Id String for this **Connection** (p. 1262).

#### Exceptions:

*CMSEException* (p. 1160) if the provider fails to return the client id or an internal error occurs.

Implements **cms::Connection** (p. 1264).

### 6.23.3.12 `unsigned int activemq::core::ActiveMQConnection::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

#### Returns:

the close timeout configured in the connection uri

### 6.23.3.13 `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId () const throw ( exceptions::ActiveMQException )`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

### 6.23.3.14 `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo () const throw ( exceptions::ActiveMQException )`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

### 6.23.3.15 `virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener () const [virtual]`

Gets the registered Exception Listener for this connection.

#### Returns:

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 1264).

**6.23.3.16** `virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const throw ( cms::CMSException ) [inline, virtual]`

Gets the metadata for this connection.

**Returns:**

the connection MetaData pointer ( caller does not own it ).

**Exceptions:**

*CMSException* if the provider fails to get the connection metadata for this connection.

**See also:**

ConnectionMetaData

**Since:**

2.0

Implements **cms::Connection** (p. 1265).

**6.23.3.17** `long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ()`

Get the Next Temporary Destination Id.

**Returns:**

the next id in the sequence.

**6.23.3.18** `long long activemq::core::ActiveMQConnection::getNextSessionId ()`

Get the Next available Session Id.

**Returns:**

the next id in the sequence.

**6.23.3.19** `long long activemq::core::ActiveMQConnection::getNextTempDestinationId ()`

Get the Next Temporary Destination Id.

**Returns:**

the next id in the sequence.



**6.23.3.20** `const std::string& activemq::core::ActiveMQConnection::getPassword () const`

Gets the password that this factory will use when creating a new connection instance.

**Returns:**

password string, "" for default credentials

**6.23.3.21** `PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy () const`

Gets the pointer to the current **PrefetchPolicy** (p. 2976) that is in use by this ConnectionFactory.

**Returns:**

a pointer to this objects **PrefetchPolicy** (p. 2976).

**6.23.3.22** `unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize () const`

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

**Returns:**

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

**6.23.3.23** `RedeliveryPolicy* activemq::core::ActiveMQConnection::getRedeliveryPolicy () const`

Gets the pointer to the current **RedeliveryPolicy** (p. 3177) that is in use by this ConnectionFactory.

**Returns:**

a pointer to this objects **RedeliveryPolicy** (p. 3177).

**6.23.3.24** `unsigned int activemq::core::ActiveMQConnection::getSendTimeout () const`

Gets the assigned send timeout for this Connector.

**Returns:**

the send timeout configured in the connection uri

**6.23.3.25** `transport::Transport& activemq::core::ActiveMQConnection::getTransport () const`

Gets a reference to this object's Transport instance.

**Returns:**

a reference to the Transport that is in use by this Connection.

**6.23.3.26** `const std::string& activemq::core::ActiveMQConnection::getUsername () const`

Gets the username that this factory will use when creating a new connection instance.

**Returns:**

username string, "" for default credentials

**6.23.3.27** `bool activemq::core::ActiveMQConnection::isAlwaysSyncSend () const`

Gets if the Connection should always send things Synchronously.

**Returns:**

true if sends should always be Synchronous.

**6.23.3.28** `bool activemq::core::ActiveMQConnection::isClosed () const [inline]`

Checks if this connection has been closed.

**Returns:**

true if the connection is closed

**6.23.3.29** `bool activemq::core::ActiveMQConnection::isDispatchAsync () const`

**Returns:**

The value of the dispatch asynchronously option sent to the broker.

**6.23.3.30** `bool activemq::core::ActiveMQConnection::isStarted () const [inline]`

Check if this connection has been started.

**Returns:**

true if the start method has been called.

**6.23.3.31** `bool activemq::core::ActiveMQConnection::isTransportFailed () const`  
[inline]

Checks if the Connection's Transport has failed.

**Returns:**

true if the Connection's Transport has failed.

**6.23.3.32** `bool activemq::core::ActiveMQConnection::isUseAsyncSend () const`

Gets if the useAsyncSend option is set.

**Returns:**

true if on false if not.

**6.23.3.33** `bool activemq::core::ActiveMQConnection::isUseCompression () const`

Gets if the Connection is configured for Message body compression.

**Returns:**

if the Message body will be Compressed or not.

**6.23.3.34** `virtual void activemq::core::ActiveMQConnection::onCommand (const  
Pointer< commands::Command > & command)` [virtual]

Event handler for the receipt of a non-response command from the **transport** (p. 97).

**Parameters:**

*command* the received command object.

Implements `activemq::transport::TransportListener` (p. 3900).

**6.23.3.35** `void activemq::core::ActiveMQConnection::oneway  
(Pointer< commands::Command > command) throw (  
activemq::exceptions::ActiveMQException )`

Sends a oneway message.

**Parameters:**

*command* The message to send.

**Exceptions:**

*ConnectorException* if not currently connected, or if the operation fails for any reason.

**6.23.3.36** `virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command **transport** (p. 97).

**Parameters:**

*ex* The exception.

Implements **activemq::transport::TransportListener** (p. 3901).

**6.23.3.37** `virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & consumer) throw ( cms::CMSEException ) [virtual]`

Removes the dispatcher for a consumer.

**Parameters:**

*consumer* - The consumer for which to remove the dispatcher.

**6.23.3.38** `virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & producerId) throw ( cms::CMSEException ) [virtual]`

Removes an active Producer to the Set of known producers.

**Parameters:**

*producerId* - The ProducerId to remove from the the known set.

**6.23.3.39** `virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * session) throw ( cms::CMSEException ) [virtual]`

Removes the session resources for the given session instance.

**Parameters:**

*session* The session to be unregistered from this connection.

**6.23.3.40** `void activemq::core::ActiveMQConnection::removeTransportListener (transport::TransportListener * transportListener)`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events. The caller is responsible for freeing the listener in all cases.

**Parameters:**

*transportListener* The pointer to the TransportListener to remove from the set of listeners.

**6.23.3.41** `virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * consumer, long long timeout) throw (exceptions::ActiveMQException) [virtual]`

If supported sends a message pull request to the service provider asking for the delivery of a new message. This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

**Parameters:**

*consumer* - the ConsumerInfo for the requesting Consumer.

*timeout* - the time that the client is willing to wait.

**6.23.3.42** `void activemq::core::ActiveMQConnection::setAlwaysSyncSend (bool value)`

Sets if the Connection should always send things Synchronously.

**Parameters:**

*value* true if sends should always be Synchronous.

**6.23.3.43** `void activemq::core::ActiveMQConnection::setBrokerURL (const std::string & brokerURL)`

Sets the Broker URL that should be used when creating a new connection instance.

**Parameters:**

*brokerURL* string

**6.23.3.44** `virtual void activemq::core::ActiveMQConnection::setClientID (const std::string & clientID) [virtual]`

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p.1324) object and transparently assigned to the **Connection** (p.1262) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p.1997).

**Parameters:**

*clientID* The unique client identifier to assign to the **Connection** (p.1262).

**Exceptions:**

**CMSException** (p. 1160) if the provider fails to set the client id due to some internal error.

**InvalidClientIDException** if the id given is somehow invalid or is a duplicate.

*IllegalStateException* (p. 1997) if the client tries to set the id after a **Connection** (p. 1262) method has been called.

Implements **cms::Connection** (p. 1265).

#### 6.23.3.45 void activemq::core::ActiveMQConnection::setCloseTimeout (unsigned int *timeout*)

Sets the close timeout to use when sending the disconnect request.

##### Parameters:

*timeout* - The time to wait for a close message.

#### 6.23.3.46 void activemq::core::ActiveMQConnection::setDefaultClientId (const std::string & *clientId*)

Sets the Client Id.

##### Parameters:

*clientId* - The new clientId value.

#### 6.23.3.47 void activemq::core::ActiveMQConnection::setDispatchAsync (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false. For slow consumers set it to true so that dispatching will not block fast consumers. .

##### Parameters:

*value* The value of the dispatch asynchronously option sent to the broker.

#### 6.23.3.48 virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener \* *listener*) [virtual]

Sets the registered Exception Listener for this connection.

##### Parameters:

*listener* pointer to and ExceptionListener

Implements **cms::Connection** (p. 1265).

#### 6.23.3.49 void activemq::core::ActiveMQConnection::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

##### Parameters:

*password* string

**6.23.3.50 void activemq::core::ActiveMQConnection::setPrefetchPolicy**  
(PrefetchPolicy \* *policy*)

Sets the **PrefetchPolicy** (p. 2976) instance that this factory should use when it creates new Connection instances. The **PrefetchPolicy** (p. 2976) passed becomes the property of the factory and will be deleted when the factory is destroyed.

**Parameters:**

*policy* The new **PrefetchPolicy** (p. 2976) that the ConnectionFactory should clone for Connections.

**6.23.3.51 void activemq::core::ActiveMQConnection::setProducerWindowSize**  
(unsigned int *windowSize*)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

**Parameters:**

*windowSize* - The size in bytes of the Producers memory window.

**6.23.3.52 void activemq::core::ActiveMQConnection::setRedeliveryPolicy**  
(RedeliveryPolicy \* *policy*)

Sets the **RedeliveryPolicy** (p. 3177) instance that this factory should use when it creates new Connection instances. The **RedeliveryPolicy** (p. 3177) passed becomes the property of the factory and will be deleted when the factory is destroyed.

**Parameters:**

*policy* The new **RedeliveryPolicy** (p. 3177) that the ConnectionFactory should clone for Connections.

**6.23.3.53 void activemq::core::ActiveMQConnection::setSendTimeout** (unsigned int *timeout*)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

**Parameters:**

*timeout* - The time to wait for a response.

**6.23.3.54 void activemq::core::ActiveMQConnection::setTransportInterruptProcessingComplete**  
( )

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

**6.23.3.55** `void activemq::core::ActiveMQConnection::setUseAsyncSend (bool value)`

Sets the useAsyncSend option.

**Parameters:**

*value* - true to activate, false to disable.

**6.23.3.56** `void activemq::core::ActiveMQConnection::setUseCompression (bool value)`

Sets whether Message body compression is enabled.

**Parameters:**

*value* Boolean indicating if Message body compression is enabled.

**6.23.3.57** `void activemq::core::ActiveMQConnection::setUsername (const std::string & username)`

Sets the username that should be used when creating a new connection.

**Parameters:**

*username* string

**6.23.3.58** `virtual void activemq::core::ActiveMQConnection::start () throw (cms::CMSEException ) [virtual]`

Starts or (restarts) a connections delivery of incoming messages.

**Exceptions:**

*CMSEException*

Implements `cms::Startable` (p. 3586).

**6.23.3.59** `virtual void activemq::core::ActiveMQConnection::stop () throw (cms::CMSEException ) [virtual]`

Stop the flow of incoming messages.

**Exceptions:**

*CMSEException*

Implements `cms::Stoppable` (p. 3648).



**6.23.3.60**    `void activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw ( activemq::exceptions::ActiveMQException )`

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

**Parameters:**

*command*    The request command.

*timeout*    The time to wait for a response, default is zero or infinite.

**Exceptions:**

*ConnectorException* thrown if an error response was received from the broker, or if any other error occurred.

**6.23.3.61**    `virtual void activemq::core::ActiveMQConnection::transportInterrupted () [virtual]`

The **transport** (p.97) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p.3901).

**6.23.3.62**    `virtual void activemq::core::ActiveMQConnection::transportResumed () [virtual]`

The **transport** (p.97) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p.3901).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnection.h`

## 6.24 activemq::core::ActiveMQConnectionFactory Class Reference

#include <src/main/activemq/core/ActiveMQConnectionFactory.h> Inheritance diagram for activemq::core::ActiveMQConnectionFactory:

### Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")  
*Constructor.*
- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection \* createConnection** () throw ( cms::CMSEException )  
*Creates a connection with the default user identity.*
- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password) throw ( cms::CMSEException )  
*Creates a connection with the specified user identity.*
- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw ( cms::CMSEException )  
*Creates a connection with the specified user identity.*
- void **setUsername** (const std::string &username)  
*Sets the username that should be used when creating a new connection.*
- const std::string & **getUsername** () const  
*Gets the username that this factory will use when creating a new connection instance.*
- void **setPassword** (const std::string &password)  
*Sets the password that should be used when creating a new connection.*
- const std::string & **getPassword** () const  
*Gets the password that this factory will use when creating a new connection instance.*
- std::string **getClientId** () const  
*Gets the Configured Client Id.*
- void **setClientId** (const std::string &clientId)  
*Sets the Client Id.*
- void **setBrokerURL** (const std::string &brokerURL)  
*Sets the Broker URL that should be used when creating a new connection instance.*
- const std::string & **getBrokerURL** () const

*Gets the Broker URL that this factory will use when creating a new connection instance.*

- **void setExceptionListener (cms::ExceptionListener \*listener)**  
*Set an CMS ExceptionListener that will be set on eat connection once it has been created.*
- **cms::ExceptionListener \* getExceptionListener () const**  
*Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.*
- **void setPrefetchPolicy (PrefetchPolicy \*policy)**  
*Sets the **PrefetchPolicy** (p. 2976) instance that this factory should use when it creates new Connection instances.*
- **PrefetchPolicy \* getPrefetchPolicy () const**  
*Gets the pointer to the current **PrefetchPolicy** (p. 2976) that is in use by this ConnectionFactory.*
- **void setRedeliveryPolicy (RedeliveryPolicy \*policy)**  
*Sets the **RedeliveryPolicy** (p. 3177) instance that this factory should use when it creates new Connection instances.*
- **RedeliveryPolicy \* getRedeliveryPolicy () const**  
*Gets the pointer to the current **RedeliveryPolicy** (p. 3177) that is in use by this ConnectionFactory.*
- **bool isDispatchAsync () const**
- **void setDispatchAsync (bool value)**  
*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*
- **bool isAlwaysSyncSend () const**  
*Gets if the Connection should always send things Synchronously.*
- **void setAlwaysSyncSend (bool value)**  
*Sets if the Connection should always send things Synchronously.*
- **bool isUseAsyncSend () const**  
*Gets if the useAsyncSend option is set.*
- **void setUseAsyncSend (bool value)**  
*Sets the useAsyncSend option.*
- **bool isUseCompression () const**  
*Gets if the Connection is configured for Message body compression.*
- **void setUseCompression (bool value)**  
*Sets whether Message body compression is enabled.*
- **unsigned int getSendTimeout () const**  
*Gets the assigned send timeout for this Connector.*

- void **setSendTimeout** (unsigned int timeout)  
*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*
- unsigned int **getCloseTimeout** () const  
*Gets the assigned close timeout for this Connector.*
- void **setCloseTimeout** (unsigned int timeout)  
*Sets the close timeout to use when sending the disconnect request.*
- unsigned int **getProducerWindowSize** () const  
*Gets the configured producer window size for Producers that are created from this connector.*
- void **setProducerWindowSize** (unsigned int windowSize)  
*Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.*

## Static Public Member Functions

- static **cms::Connection \* createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSEException )  
*Creates a connection with the specified user identity.*

## Static Public Attributes

- static const std::string **DEFAULT\_URI**

### 6.24.1 Constructor & Destructor Documentation

- 6.24.1.1 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory** ()
- 6.24.1.2 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory** (const std::string & *url*, const std::string & *username* = "", const std::string & *password* = "")

Constructor.

#### Parameters:

- url* the URL of the Broker we are connecting to.
- username* to authenticate with, defaults to ""
- password* to authenticate with, defaults to ""

**6.24.1.3**    **virtual**  
              **activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory**  
              **()**    [virtual]

## 6.24.2 Member Function Documentation

**6.24.2.1**    **static cms::Connection\* ac-**  
              **tivemq::core::ActiveMQConnectionFactory::createConnection (const**  
              **std::string & url, const std::string & username, const std::string &**  
              **password, const std::string & clientId = "") throw ( cms::CMSException**  
              **)**    [static]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

### Parameters:

*url* the URL of the Broker we are connecting to.  
*username* to authenticate with  
*password* to authenticate with  
*clientId* to assign to connection, defaults to ""

### Exceptions:

*CMSException*.

**6.24.2.2**    **virtual cms::Connection\* ac-**  
              **tivemq::core::ActiveMQConnectionFactory::createConnection (const**  
              **std::string & username, const std::string & password, const std::string &**  
              **clientId) throw ( cms::CMSException )**    [virtual]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

### Parameters:

*username* to authenticate with  
*password* to authenticate with  
*clientId* to assign to connection if "" then a random client Id is created for this connection.

### Returns:

a Connection Pointer

### Exceptions:

*CMSException*

Implements **cms::ConnectionFactory** (p. 1325).

**6.24.2.3** `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password) throw ( cms::CMSEException )` [virtual]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

**Parameters:**

*username* to authenticate with

*password* to authenticate with

**Returns:**

a Connection Pointer

**Exceptions:**

*CMSEException*

Implements **cms::ConnectionFactory** (p. 1326).

**6.24.2.4** `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection () throw ( cms::CMSEException )` [virtual]

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

**Returns:**

a Connection Pointer

**Exceptions:**

*CMSEException*

Implements **cms::ConnectionFactory** (p. 1326).

**6.24.2.5** `const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL () const`

Gets the Broker URL that this factory will use when creating a new connection instance.

**Returns:**

brokerURL string

**6.24.2.6** `std::string activemq::core::ActiveMQConnectionFactory::getClientId () const`

Gets the Configured Client Id.

**Returns:**

the clientId.

**6.24.2.7** `unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout () const`

Gets the assigned close timeout for this Connector.

**Returns:**

the close timeout configured in the connection uri

**6.24.2.8** `cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener () const`

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

**Returns:**

a pointer to a CMS ExceptionListener instance or NULL if not set.

**6.24.2.9** `const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const`

Gets the password that this factory will use when creating a new connection instance.

**Returns:**

password string, "" for default credentials

**6.24.2.10** `PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy () const`

Gets the pointer to the current **PrefetchPolicy** (p. 2976) that is in use by this ConnectionFactory.

**Returns:**

a pointer to this objects **PrefetchPolicy** (p. 2976).

**6.24.2.11** `unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize () const`

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

**Returns:**

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

**6.24.2.12** `RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy () const`

Gets the pointer to the current **RedeliveryPolicy** (p. 3177) that is in use by this ConnectionFactory.

**Returns:**

a pointer to this objects **RedeliveryPolicy** (p. 3177).

**6.24.2.13** `unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout () const`

Gets the assigned send timeout for this Connector.

**Returns:**

the send timeout configured in the connection uri

**6.24.2.14** `const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const`

Gets the username that this factory will use when creating a new connection instance.

**Returns:**

username string, "" for default credentials

**6.24.2.15** `bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend () const`

Gets if the Connection should always send things Synchronously.

**Returns:**

true if sends should always be Synchronous.



**6.24.2.16**   `bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync ()`  
                  `const`

**Returns:**

The value of the dispatch asynchronously option sent to the broker.

**6.24.2.17**   `bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend ()`  
                  `const`

Gets if the useAsyncSend option is set.

**Returns:**

true if on false if not.

**6.24.2.18**   `bool activemq::core::ActiveMQConnectionFactory::isUseCompression ()`  
                  `const`

Gets if the Connection is configured for Message body compression.

**Returns:**

if the Message body will be Compressed or not.

**6.24.2.19**   `void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend`  
                  `(bool value)`

Sets if the Connection should always send things Synchronously.

**Parameters:**

*value* true if sends should always be Synchronous.

**6.24.2.20**   `void activemq::core::ActiveMQConnectionFactory::setBrokerURL (const`  
                  `std::string & brokerURL)`

Sets the Broker URL that should be used when creating a new connection instance.

**Parameters:**

*brokerURL* string

**6.24.2.21**   `void activemq::core::ActiveMQConnectionFactory::setClientId (const`  
                  `std::string & clientId)`

Sets the Client Id.

**Parameters:**

*clientId* - The new clientId value.

#### 6.24.2.22 void activemq::core::ActiveMQConnectionFactory::setCloseTimeout (unsigned int *timeout*)

Sets the close timeout to use when sending the disconnect request.

##### Parameters:

*timeout* - The time to wait for a close message.

#### 6.24.2.23 void activemq::core::ActiveMQConnectionFactory::setDispatchAsync (bool *value*)

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false. For slow consumers set it to true so that dispatching will not block fast consumers. .

##### Parameters:

*value* The value of the dispatch asynchronously option sent to the broker.

#### 6.24.2.24 void activemq::core::ActiveMQConnectionFactory::setExceptionListener (cms::ExceptionListener \* *listener*)

Set an CMS ExceptionListener that will be set on eat connection once it has been created. The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

##### Parameters:

*listener* The listener to set on the connection or NULL for no listener.

#### 6.24.2.25 void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & *password*)

Sets the password that should be used when creating a new connection.

##### Parameters:

*password* string

#### 6.24.2.26 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy (PrefetchPolicy \* *policy*)

Sets the **PrefetchPolicy** (p.2976) instance that this factory should use when it creates new Connection instances. The **PrefetchPolicy** (p.2976) passed becomes the property of the factory and will be deleted when the factory is destroyed.

##### Parameters:

*policy* The new **PrefetchPolicy** (p.2976) that the ConnectionFactory should clone for Connections.

#### 6.24.2.27 void activemq::core::ActiveMQConnectionFactory::setProducerWindowSize (unsigned int *windowSize*)

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

##### Parameters:

*windowSize* - The size in bytes of the Producers memory window.

#### 6.24.2.28 void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy (RedeliveryPolicy \* *policy*)

Sets the **RedeliveryPolicy** (p. 3177) instance that this factory should use when it creates new Connection instances. The **RedeliveryPolicy** (p. 3177) passed becomes the property of the factory and will be deleted when the factory is destroyed.

##### Parameters:

*policy* The new **RedeliveryPolicy** (p. 3177) that the ConnectionFactory should clone for Connections.

#### 6.24.2.29 void activemq::core::ActiveMQConnectionFactory::setSendTimeout (unsigned int *timeout*)

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

##### Parameters:

*timeout* - The time to wait for a response.

#### 6.24.2.30 void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend (bool *value*)

Sets the useAsyncSend option.

##### Parameters:

*value* - true to activate, false to disable.

#### 6.24.2.31 void activemq::core::ActiveMQConnectionFactory::setUseCompression (bool *value*)

Sets whether Message body compression is enabled.

##### Parameters:

*value* Boolean indicating if Message body compression is enabled.

**6.24.2.32**   `void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & username)`

Sets the username that should be used when creating a new connection.

**Parameters:**

*username* string

### 6.24.3   Field Documentation

**6.24.3.1**   `const std::string  
activemq::core::ActiveMQConnectionFactory::DEFAULT_ -  
URI   [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionFactory.h`

## 6.25 activemq::core::ActiveMQConnectionMetaData Class Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 273) class.

#include <src/main/activemq/core/ActiveMQConnectionMetaData.h> Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

### Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual **~ActiveMQConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const throw ( cms::CMSEException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS minor version number.*
- virtual std::string **getCMSProviderName** () const throw ( cms::CMSEException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider major version number.*
- virtual int **getProviderMinorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw ( cms::CMSEException )  
*Gets an Vector of the CMSX property names.*

### 6.25.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 273) class.

Since:

3.0

## 6.25.2 Constructor & Destructor Documentation

**6.25.2.1** `activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData()`

**6.25.2.2** `virtual activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData()` [virtual]

## 6.25.3 Member Function Documentation

**6.25.3.1** `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion()`  
`const throw ( cms::CMSEException )` [virtual]

Gets the CMS major version number.

### Returns:

the CMS API major version number

### Exceptions:

***CMSEException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1386).

**6.25.3.2** `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion()`  
`const throw ( cms::CMSEException )` [virtual]

Gets the CMS minor version number.

### Returns:

the CMS API minor version number

### Exceptions:

***CMSEException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1386).

**6.25.3.3** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName()`  
`const throw ( cms::CMSEException )` [virtual]

Gets the CMS provider name.

### Returns:

the CMS provider name

**Exceptions:**

***CMSException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1386).

**6.25.3.4** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const throw ( cms::CMSException ) [virtual]`

Gets the CMS API version.

**Returns:**

the CMS API Version in String form.

**Exceptions:**

***CMSException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1387).

**6.25.3.5** `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const throw ( cms::CMSException ) [virtual]`

Gets an Vector of the CMSX property names.

**Returns:**

an Vector of CMSX property names

**Exceptions:**

***CMSException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1387).

**6.25.3.6** `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider major version number.

**Returns:**

the CMS provider major version number

**Exceptions:**

***CMSException*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1387).

**6.25.3.7** `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider minor version number.

**Returns:**

the CMS provider minor version number

**Exceptions:**

*CMSException* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1387).

**6.25.3.8** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider version.

**Returns:**

the CMS provider version

**Exceptions:**

*CMSException* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1388).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionMetaData.h`



## 6.26 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

### Data Structures

- class `StaticInitializer`

### Public Types

- enum `TransactionState` {  
`TRANSACTION_STATE_BEGIN` = 0, `TRANSACTION_STATE_PREPARE` = 1, `TRANSACTION_STATE_COMMITONEPHASE` = 2, `TRANSACTION_STATE_COMMITTWOPHASE` = 3,  
`TRANSACTION_STATE_ROLLBACK` = 4, `TRANSACTION_STATE_RECOVER` = 5, `TRANSACTION_STATE_FORGET` = 6, `TRANSACTION_STATE_END` = 7 }
  - enum `DestinationActions` { `DESTINATION_ADD_OPERATION` = 0, `DESTINATION_REMOVE_OPERATION` = 1 }
  - enum `AckType` {  
`ACK_TYPE_DELIVERED` = 0, `ACK_TYPE_POISON` = 1, `ACK_TYPE_CONSUMED` = 2, `ACK_TYPE_REDELIVERED` = 3,  
`ACK_TYPE_INDIVIDUAL` = 4 }
  - enum `DestinationOption` {  
`CONSUMER_PREFETCHSIZE`, `CONSUMER_MAXPENDINGMSGLIMIT`, `CONSUMER_NOLOCAL`, `CONSUMER_DISPATCHASYNC`,  
`CONSUMER_RETROACTIVE`, `CONSUMER_SELECTOR`, `CONSUMER_EXCLUSIVE`, `CONSUMER_PRIORITY`,  
`NUM_OPTIONS` }
- These values represent the options that can be appended to an Destination name, i.e.*
- enum `URIParam` {  
`CONNECTION_SENDTIMEOUT`, `CONNECTION_PRODUCERWINDOWSIZE`, `CONNECTION_CLOSETIMEOUT`,  
`CONNECTION_ALWAYSSENDCONNECTION_USEASYNCSEND`, `CONNECTION_USECOMPRESSION`,  
`CONNECTION_DISPATCHASYNC`, `PARAM_USERNAME`,  
`PARAM_PASSWORD`, `PARAM_CLIENTID`, `NUM_PARAMS` }
- These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.*

## Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam** **toURIOption** (const std::string &option)

### 6.26.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

### 6.26.2 Member Enumeration Documentation

#### 6.26.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

```
ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL
```

#### 6.26.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

```
DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION
```

#### 6.26.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e. /topic/-foo?consumer.exclusive=true

Enumerator:

```
CONSUMER_PREFETCHSIZE
CUNSUMER_MAXPENDINGMSGLIMIT
CONSUMER_NOLOCAL
CONSUMER_DISPATCHASYNC
CONSUMER_RETROACTIVE
CONSUMER_SELECTOR
CONSUMER_EXCLUSIVE
CONSUMER_PRIORITY
NUM_OPTIONS
```

#### 6.26.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

```
TRANSACTION_STATE_BEGIN
TRANSACTION_STATE_PREPARE
TRANSACTION_STATE_COMMITONEPHASE
TRANSACTION_STATE_COMMITTWOPHASE
TRANSACTION_STATE_ROLLBACK
TRANSACTION_STATE_RECOVER
TRANSACTION_STATE_FORGET
TRANSACTION_STATE_END
```

#### 6.26.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

```
CONNECTION_SENDTIMEOUT
CONNECTION_PRODUCERWINDOWSIZE
CONNECTION_CLOSETIMEOUT
CONNECTION_ALWAYS_SYNC_SEND
CONNECTION_USE_ASYNC_SEND
CONNECTION_USE_COMPRESSION
CONNECTION_DISPATCH_ASYNC
PARAM_USERNAME
PARAM_PASSWORD
PARAM_CLIENTID
NUM_PARAMS
```

### 6.26.3 Member Function Documentation

**6.26.3.1** static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption  
(const std::string & option) [inline, static]

**6.26.3.2** static const std::string& activemq::core::ActiveMQConstants::toString  
(const URIParam option) [inline, static]

**6.26.3.3** static const std::string& activemq::core::ActiveMQConstants::toString  
(const DestinationOption option) [inline, static]

**6.26.3.4** static URIParam activemq::core::ActiveMQConstants::toURIOption  
(const std::string & option) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConstants.h

## 6.27 activemq::core::ActiveMQConsumer Class Reference

#include <src/main/activemq/core/ActiveMQConsumer.h> Inheritance diagram for activemq::core::ActiveMQConsumer:

### Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** \*session, const **Pointer**< **commands::ConsumerId** > &id, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &name, const std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** \*listener)  
*Constructor.*
- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()  
*Starts the Consumer if not already started and not closed.*
- virtual void **stop** ()  
*Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.*
- virtual void **close** () throw ( cms::CMSException )  
*Closes the Consumer.*
- virtual **cms::Message** \* **receive** () throw ( cms::CMSException )  
*Synchronously Receive a Message.*
- virtual **cms::Message** \* **receive** (int millisecs) throw ( cms::CMSException )  
*Synchronously Receive a Message, time out after defined interval.*
- virtual **cms::Message** \* **receiveNoWait** () throw ( cms::CMSException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**cms::MessageListener** \*listener) throw ( cms::CMSException )  
*Sets the MessageListener that this class will send notifs on.*
- virtual **cms::MessageListener** \* **getMessageListener** () const throw ( cms::CMSException )  
*Gets the MessageListener that this class will send events to.*
- virtual std::string **getMessageSelector** () const throw ( cms::CMSException )  
*Gets this message consumer's message selector expression.*
- virtual void **acknowledge** (const **Pointer**< **commands::MessageDispatch** > &dispatch) throw ( cms::CMSException )

*Method called to acknowledge the message passed, called from a message when the mode is client ack.*

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)  
*Called asynchronously by the session to dispatch a message.*
- void **acknowledge** () throw ( cms::CMSException )  
*Method called to acknowledge all messages that have been received so far.*
- void **commit** () throw ( exceptions::ActiveMQException )  
*Called to Commit the current set of messages in this Transaction.*
- void **rollback** () throw ( exceptions::ActiveMQException )  
*Called to Roll back the current set of messages in this Transaction.*
- void **doClose** () throw ( exceptions::ActiveMQException )  
*Performs the actual close operation on this consumer.*
- const **Pointer**< **commands::ConsumerInfo** > & **getConsumerInfo** () const  
*Get the Consumer information for this consumer.*
- const **Pointer**< **commands::ConsumerId** > & **getConsumerId** () const  
*Get the Consumer Id for this consumer.*
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const  
*Has this Consumer Transaction **Synchronization** (p. 3715) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)  
*Sets the **Synchronization** (p. 3715) Registered **state** (p. 95) of this consumer.*
- bool **iterate** ()  
*Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.*
- void **deliverAcks** () throw ( exceptions::ActiveMQException )  
*Forces this consumer to send all pending acks to the broker.*
- void **clearMessagesInProgress** ()  
*Called on a Failover to clear any pending messages.*
- void **inProgressClearRequired** ()  
*Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.*
- long long **getLastDeliveredSequenceId** () const  
*Gets the currently set Last Delivered Sequence Id.*
- void **setLastDeliveredSequenceId** (long long value)  
*Sets the value of the Last Delivered Sequence Id.*

- `int getMessageAvailableCount () const`
- `void setRedeliveryPolicy (RedeliveryPolicy *policy)`  
*Sets the **RedeliveryPolicy** (p. 3177) this Consumer should use when a rollback is performed on a transacted Consumer.*
- `RedeliveryPolicy * getRedeliveryPolicy () const`  
*Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.*

## Protected Member Functions

- `Pointer< MessageDispatch > dequeue (long long timeout) throw ( cms::CMSException )`  
*Used by synchronous receive methods to wait for messages to come in.*
- `void beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > &dispatch)`  
*Pre-consume processing.*
- `void afterMessageIsConsumed (const Pointer< commands::MessageDispatch > &dispatch, bool messageExpired)`  
*Post-consume processing.*

## 6.27.1 Constructor & Destructor Documentation

- 6.27.1.1 `activemq::core::ActiveMQConsumer::ActiveMQConsumer (ActiveMQSession * session, const Pointer< commands::ConsumerId > & id, const Pointer< commands::ActiveMQDestination > & destination, const std::string & name, const std::string & selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, cms::MessageListener * listener)`

Constructor.

- 6.27.1.2 `virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer ()`  
`[virtual]`

## 6.27.2 Member Function Documentation

- 6.27.2.1 `void activemq::core::ActiveMQConsumer::acknowledge () throw ( cms::CMSException )`

Method called to acknowledge all messages that have been received so far.

Exceptions:

*CMSException*

**6.27.2.2** `virtual void activemq::core::ActiveMQConsumer::acknowledge (const Pointer< commands::MessageDispatch > & dispatch) throw ( cms::CMSException ) [virtual]`

Method called to acknowledge the message passed, called from a message when the mode is client ack.

**Parameters:**

*message* the Message to Acknowledge

**Exceptions:**

*CMSException*

**6.27.2.3** `void activemq::core::ActiveMQConsumer::afterMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch, bool messageExpired) [protected]`

Post-consume processing.

**Parameters:**

*dispatch* - the consumed message

*messageExpired* - flag indicating if the message has expired.

**6.27.2.4** `void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch) [protected]`

Pre-consume processing.

**Parameters:**

*dispatch* - the message being consumed.

**6.27.2.5** `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()`

Called on a Failover to clear any pending messages.

**6.27.2.6** `virtual void activemq::core::ActiveMQConsumer::close () throw ( cms::CMSException ) [virtual]`

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

**Exceptions:**

*CMSException*

Implements `cms::Closeable` (p.1148).

**6.27.2.7** `void activemq::core::ActiveMQConsumer::commit () throw ( exceptions::ActiveMQException )`

Called to Commit the current set of messages in this Transaction.

**Exceptions:**

*ActiveMQException*

**6.27.2.8** `void activemq::core::ActiveMQConsumer::deliverAcks () throw ( exceptions::ActiveMQException )`

Forces this consumer to send all pending acks to the broker.

**6.27.2.9** `Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long timeout) throw ( cms::CMSException ) [protected]`

Used by synchronous receive methods to wait for messages to come in.

**Parameters:**

*timeout* - The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

**Returns:**

the message, if received within the allotted time. Otherwise NULL.

**Exceptions:**

*InvalidStateException* if this consumer is closed upon entering this method.

**6.27.2.10** `virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & message) [virtual]`

Called asynchronously by the session to dispatch a message.

**Parameters:**

*message* dispatch object pointer

Implements `activemq::core::Dispatcher` (p. 1787).

**6.27.2.11** `void activemq::core::ActiveMQConsumer::doClose () throw ( exceptions::ActiveMQException )`

Performs the actual close operation on this consumer.

**Exceptions:**

*ActiveMQException*



**6.27.2.12** `const Pointer<commands::ConsumerId>& activemq::core::ActiveMQConsumer::getConsumerId () const [inline]`

Get the Consumer Id for this consumer.

**Returns:**

Reference to a Consumer Id Object

**6.27.2.13** `const Pointer<commands::ConsumerInfo>& activemq::core::ActiveMQConsumer::getConsumerInfo () const [inline]`

Get the Consumer information for this consumer.

**Returns:**

Reference to a Consumer Info Object

**6.27.2.14** `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

**Returns:**

long long containing the sequence id of the last delivered Message.

**6.27.2.15** `int activemq::core::ActiveMQConsumer::getMessageAvailableCount () const`

**Returns:**

the number of Message's this consumer is waiting to Dispatch.

**6.27.2.16** `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener () const throw ( cms::CMSException ) [inline, virtual]`

Gets the MessageListener that this class will send events to.

**Returns:**

the currently registered MessageListener interface pointer.

Implements `cms::MessageConsumer` (p. 2590).

**6.27.2.17** `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector ()  
const throw ( cms::CMSException ) [virtual]`

Gets this message consumer's message selector expression.

**Returns:**

This Consumer's selector expression or "".

**Exceptions:**

*cms::CMSException* (p. 1160)

Implements `cms::MessageConsumer` (p. 2590).

**6.27.2.18** `RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy ()  
const [inline]`

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

**Returns:**

a Pointer to a **RedeliveryPolicy** (p. 3177) that is in use by this Consumer.

**6.27.2.19** `void activemq::core::ActiveMQConsumer::inProgressClearRequired ()`

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

**6.27.2.20** `bool activemq::core::ActiveMQConsumer::isClosed () const [inline]`

**Returns:**

if this Consumer has been closed.

**6.27.2.21** `bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered ()  
const [inline]`

Has this Consumer Transaction **Synchronization** (p. 3715) been added to the transaction.

**Returns:**

true if the synchronization has been added.

**6.27.2.22** `bool activemq::core::ActiveMQConsumer::iterate ()`

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

**6.27.2.23** `virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int milliseconds) throw ( cms::CMSException ) [virtual]`

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

**Parameters:**

*milliseconds* the time in milliseconds to wait before returning

**Returns:**

new message or null on timeout

**Exceptions:**

*CMSException*

Implements `cms::MessageConsumer` (p. 2590).

**6.27.2.24** `virtual cms::Message* activemq::core::ActiveMQConsumer::receive () throw ( cms::CMSException ) [virtual]`

Synchronously Receive a Message.

**Returns:**

new message

**Exceptions:**

*CMSException*

Implements `cms::MessageConsumer` (p. 2591).

**6.27.2.25** `virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait () throw ( cms::CMSException ) [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

**Returns:**

new message

**Exceptions:**

*CMSException*

Implements `cms::MessageConsumer` (p. 2591).

**6.27.2.26** `void activemq::core::ActiveMQConsumer::rollback () throw ( exceptions::ActiveMQException )`

Called to Roll back the current set of messages in this Transaction.

Exceptions:

*ActiveMQException*

**6.27.2.27** `void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId  
(long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters:

*value* The new value to assign to the Last Delivered Sequence Id property.

**6.27.2.28** `virtual void activemq::core::ActiveMQConsumer::setMessageListener  
(cms::MessageListener * listener) throw ( cms::CMSException )  
[virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters:

*listener* MessageListener interface pointer

Implements `cms::MessageConsumer` (p. 2591).

**6.27.2.29** `void activemq::core::ActiveMQConsumer::setRedeliveryPolicy  
(RedeliveryPolicy * policy) [inline]`

Sets the **RedeliveryPolicy** (p. 3177) this Consumer should use when a rollback is performed on a transacted Consumer. The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

Parameters:

*policy* Pointer to a Redelivery Policy object that his Consumer will use.

**6.27.2.30** `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered  
(bool value) [inline]`

Sets the **Synchronization** (p. 3715) Registered **state** (p. 95) of this consumer.

Parameters:

*value* - true if registered false otherwise.

**6.27.2.31** `virtual void activemq::core::ActiveMQConsumer::start () [virtual]`

Starts the Consumer if not already started and not closed. A consumer will no deliver messages until started.

**6.27.2.32 virtual void activemq::core::ActiveMQConsumer::stop () [virtual]**

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again. A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConsumer.h`

## 6.28 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

### Public Member Functions

- virtual `~ActiveMQCPP ()`

### Static Public Member Functions

- static void `initializeLibrary ()`  
*Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf **library** (p. 94).*
- static void `initializeLibrary (int argc, char **argv)`  
*Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf **library** (p. 94).*
- static void `shutdownLibrary ()`  
*Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.*

### Protected Member Functions

- `ActiveMQCPP ()`
- `ActiveMQCPP (const ActiveMQCPP &)`
- `ActiveMQCPP & operator= (const ActiveMQCPP &)`

### 6.28.1 Constructor & Destructor Documentation

6.28.1.1 `activemq::library::ActiveMQCPP::ActiveMQCPP () [inline, protected]`

6.28.1.2 `activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &) [protected]`

6.28.1.3 `virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP () [inline, virtual]`

### 6.28.2 Member Function Documentation

6.28.2.1 `static void activemq::library::ActiveMQCPP::initializeLibrary (int argc, char ** argv) [static]`

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf **library** (p.94). This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

#### Parameters:

*argc* - the count of arguments passed to this Process.

*argv* - the array of string arguments passed to this process.

**Exceptions:**

*runtime\_error* if an error occurs while initializing this **library** (p. 94).

**6.28.2.2 static void activemq:library::ActiveMQCPP::initializeLibrary () [static]**

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf **library** (p. 94).

**Exceptions:**

*runtime\_error* if an error occurs while initializing this **library** (p. 94).

**6.28.2.3 ActiveMQCPP& activemq:library::ActiveMQCPP::operator= (const ActiveMQCPP &) [protected]**

**6.28.2.4 static void activemq:library::ActiveMQCPP::shutdownLibrary () [static]**

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point. All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- src/main/activemq/library/**ActiveMQCPP.h**

## 6.29 activemq::commands::ActiveMQDestination Class Reference

#include <src/main/activemq/commands/ActiveMQDestination.h> Inheritance diagram for activemq::commands::ActiveMQDestination:

### Data Structures

- struct **DestinationFilter**

### Public Member Functions

- **ActiveMQDestination** ()
- **ActiveMQDestination** (const std::string &**physicalName**)
- virtual ~**ActiveMQDestination** ()
- virtual **ActiveMQDestination** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual const std::string & **getPhysicalName** () const  
*Fetch this destination's physical name.*
- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &**physicalName**)  
*Set this destination's physical name.*
- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool **advisory**)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isOrdered** () const



- virtual void **setOrdered** (bool **ordered**)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &**orderedTarget**)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0  
*Returns the Type of Destination that this object represents.*
- virtual bool **isTemporary** () const  
*Returns true if a temporary Destination.*
- virtual bool **isTopic** () const  
*Returns true if a Topic Destination.*
- virtual bool **isQueue** () const  
*Returns true if a Queue Destination.*
- virtual bool **isComposite** () const  
*Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.*
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** \* **getCMSDestination** () const

## Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)  
*Create a temporary name from the clientId.*
- static std::string **getClientId** (const **ActiveMQDestination** \*destination)  
*From a temporary destination find the clientId of the Connection that created it.*
- static **Pointer< ActiveMQDestination >** **createDestination** (int type, const std::string &name)  
*Creates a Destination given the String Name to use and a Type.*

## Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQDESTINATION** = 0

## Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

## Static Protected Attributes

- static const std::string **ADVISORY\_PREFIX**  
*prefix for Advisory message destinations*
- static const std::string **CONSUMER\_ADVISORY\_PREFIX**  
*prefix for consumer advisory destinations*
- static const std::string **PRODUCER\_ADVISORY\_PREFIX**  
*prefix for producer advisory destinations*
- static const std::string **CONNECTION\_ADVISORY\_PREFIX**  
*prefix for connection advisory destinations*
- static const std::string **DEFAULT\_ORDERED\_TARGET**  
*The default target for ordered destinations.*
- static const std::string **TEMP\_PREFIX**
- static const std::string **TEMP\_POSTFIX**
- static const std::string **COMPOSITE\_SEPARATOR**
- static const std::string **QUEUE\_QUALIFIED\_PREFIX**
- static const std::string **TOPIC\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_QUEUE\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_TOPIC\_QUALIFIED\_PREFIX**

## 6.29.1 Constructor & Destructor Documentation

**6.29.1.1** `activemq::commands::ActiveMQDestination::ActiveMQDestination ()`

**6.29.1.2** `activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & physicalName)`

**6.29.1.3** `virtual  
activemq::commands::ActiveMQDestination::~~ActiveMQDestination ()  
[inline, virtual]`

## 6.29.2 Member Function Documentation

**6.29.2.1** `virtual ActiveMQDestination* ac-  
tivemq::commands::ActiveMQDestination::cloneDataStructure () const  
[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 484),  
**activemq::commands::ActiveMQTempDestination** (p. 577), **ac-**  
**tivemq::commands::ActiveMQTempQueue** (p. 605), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 634), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 692).

**6.29.2.2 virtual void ac-**  
**tivemq::commands::ActiveMQDestination::copyDataStructure (const**  
**DataStructure \* src) [virtual]**

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 484),  
**activemq::commands::ActiveMQTempDestination** (p. 577), **ac-**  
**tivemq::commands::ActiveMQTempQueue** (p. 606), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 635), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 692).

Referenced by **activemq::commands::ActiveMQTempDestination::copyDataStructure()**.

**6.29.2.3 static Pointer<ActiveMQDestination> ac-**  
**tivemq::commands::ActiveMQDestination::createDestination (int type,**  
**const std::string & name) [static]**

Creates a Destination given the String Name to use and a Type.

**Parameters:**

*type* - The Type of Destination to Create

*name* - The Name to use in the creation of the Destination

**Returns:**

Pointer to a new **ActiveMQDestination** (p. 322) instance.

**6.29.2.4 static std::string ac-**  
**tivemq::commands::ActiveMQDestination::createTemporaryName (const**  
**std::string & clientId) [inline, static]**

Create a temporary name from the clientId.

**Parameters:**

*clientId*

**Returns:**

### 6.29.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 485),  
**activemq::commands::ActiveMQTempDestination** (p. 578), **ac-**  
**tivemq::commands::ActiveMQTempQueue** (p. 606), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 635), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 693).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **ac-**  
**tivemq::commands::ActiveMQTempDestination::equals()**.

### 6.29.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * destination)` [static]

From a temporary destination find the clientId of the Connection that created it.

#### Parameters:

*destination*

#### Returns:

the clientId or null if not a temporary destination

### 6.29.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination ()` const [inline, virtual]

#### Returns:

the **cms::Destination** (p. 1723) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 485),  
**activemq::commands::ActiveMQTempQueue** (p. 606), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 635), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 693).

### 6.29.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType ()` const [virtual]

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

#### Returns:

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1662).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 485),  
**activemq::commands::ActiveMQTempDestination** (p. 578), **ac-**  
**tivemq::commands::ActiveMQTempQueue** (p. 607), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 636), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 693).

**6.29.2.9 virtual cms::Destination::DestinationType ac-**  
**tivemq::commands::ActiveMQDestination::getDestinationType () const**  
 [pure virtual]

Returns the Type of Destination that this object represents.

**Returns:**

int type qualifier.

Implemented in **activemq::commands::ActiveMQQueue** (p. 485),  
**activemq::commands::ActiveMQTempQueue** (p. 607), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p. 636), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p. 693).

**6.29.2.10 const activemq::util::ActiveMQProperties& ac-**  
**tivemq::commands::ActiveMQDestination::getOptions () const**  
 [inline]

**Returns:**

a reference (const) to the options properties for this Destination.

**6.29.2.11 virtual std::string ac-**  
**tivemq::commands::ActiveMQDestination::getOrderedTarget () const**  
 [inline, virtual]

**Returns:**

Returns the orderedTarget.

**6.29.2.12 virtual std::string& ac-**  
**tivemq::commands::ActiveMQDestination::getPhysicalName () [inline,**  
**virtual]**

**6.29.2.13 virtual const std::string& ac-**  
**tivemq::commands::ActiveMQDestination::getPhysicalName () const**  
 [inline, virtual]

Fetch this destination's physical name.

**Returns:**

const string containing the name

**6.29.2.14** `virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const [inline, virtual]`

**Returns:**

Returns the advisory.

**6.29.2.15** `virtual bool activemq::commands::ActiveMQDestination::isComposite () const [inline, virtual]`

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

**Returns:**

true if this destination represents a collection of child destinations.

**6.29.2.16** `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory () const [inline, virtual]`

**Returns:**

true if this is a destination for Connection advisories

**6.29.2.17** `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory () const [inline, virtual]`

**Returns:**

true if this is a destination for Consumer advisories

**6.29.2.18** `virtual bool activemq::commands::ActiveMQDestination::isExclusive () const [inline, virtual]`

**Returns:**

Returns the exclusive.

**6.29.2.19** `virtual bool activemq::commands::ActiveMQDestination::isOrdered () const [inline, virtual]`

**Returns:**

Returns the ordered.

**6.29.2.20** `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory () const`  
[inline, virtual]

**Returns:**

true if this is a destination for Producer advisories

**6.29.2.21** `virtual bool activemq::commands::ActiveMQDestination::isQueue () const`  
[inline, virtual]

Returns true if a Queue Destination.

**Returns:**

true/false

**6.29.2.22** `virtual bool activemq::commands::ActiveMQDestination::isTemporary () const`  
[inline, virtual]

Returns true if a temporary Destination.

**Returns:**

true/false

References cms::Destination::TEMPORARY\_QUEUE, and cms::Destination::TEMPORARY\_TOPIC.

**6.29.2.23** `virtual bool activemq::commands::ActiveMQDestination::isTopic () const`  
[inline, virtual]

Returns true if a Topic Destination.

**Returns:**

true/false

References cms::Destination::TEMPORARY\_TOPIC, and cms::Destination::TOPIC.

**6.29.2.24** `virtual bool activemq::commands::ActiveMQDestination::isWildcard () const`  
[inline, virtual]

**Returns:**

true if the destination matches multiple possible destinations

**6.29.2.25** `virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool advisory)` [inline, virtual]

**Parameters:**

*advisory* The advisory to set.

**6.29.2.26** virtual void activemq::commands::ActiveMQDestination::setExclusive (bool *exclusive*) [inline, virtual]

**Parameters:**

*exclusive* The exclusive to set.

**6.29.2.27** virtual void activemq::commands::ActiveMQDestination::setOrdered (bool *ordered*) [inline, virtual]

**Parameters:**

*ordered* The ordered to set.

**6.29.2.28** virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & *orderedTarget*) [inline, virtual]

**Parameters:**

*orderedTarget* The orderedTarget to set.

**6.29.2.29** virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & *physicalName*) [virtual]

Set this destination's physical name.

**Returns:**

const string containing the name

**6.29.2.30** virtual std::string activemq::commands::ActiveMQDestination::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p.833).

Reimplemented in **activemq::commands::ActiveMQQueue** (p.486),  
**activemq::commands::ActiveMQTempDestination** (p.578), **ac-**  
**tivemq::commands::ActiveMQTempQueue** (p.607), **ac-**  
**tivemq::commands::ActiveMQTempTopic** (p.636), and **ac-**  
**tivemq::commands::ActiveMQTopic** (p.694).



### 6.29.3 Field Documentation

**6.29.3.1** `bool activemq::commands::ActiveMQDestination::advisory` [protected]

**6.29.3.2** `const std::string  
activemq::commands::ActiveMQDestination::ADVISORY_  
PREFIX` [static, protected]

prefix for Advisory message destinations

**6.29.3.3** `const std::string  
activemq::commands::ActiveMQDestination::COMPOSITE_  
SEPARATOR` [static, protected]

**6.29.3.4** `const std::string  
activemq::commands::ActiveMQDestination::CONNECTION_  
ADVISORY_PREFIX` [static, protected]

prefix for connection advisory destinations

**6.29.3.5** `const std::string  
activemq::commands::ActiveMQDestination::CONSUMER_  
ADVISORY_PREFIX` [static, protected]

prefix for consumer advisory destinations

**6.29.3.6** `const std::string  
activemq::commands::ActiveMQDestination::DEFAULT_  
ORDERED_TARGET` [static, protected]

The default target for ordered destinations.

**6.29.3.7** `bool activemq::commands::ActiveMQDestination::exclusive` [protected]

**6.29.3.8** `const unsigned char activemq::commands::ActiveMQDestination::ID _ - ACTIVEMQDESTINATION = 0` [static]

**6.29.3.9** `util::ActiveMQProperties activemq::commands::ActiveMQDestination::options` [protected]

**6.29.3.10** `bool activemq::commands::ActiveMQDestination::ordered` [protected]

**6.29.3.11** `std::string activemq::commands::ActiveMQDestination::orderedTarget` [protected]

**6.29.3.12** `std::string activemq::commands::ActiveMQDestination::physicalName` [protected]

**6.29.3.13** `const std::string activemq::commands::ActiveMQDestination::PRODUCER _ - ADVISORY _ PREFIX` [static, protected]

prefix for producer advisory destinations

**6.29.3.14** `const std::string activemq::commands::ActiveMQDestination::QUEUE _ - QUALIFIED _ PREFIX` [static, protected]

**6.29.3.15** `const std::string activemq::commands::ActiveMQDestination::TEMP _ - POSTFIX` [static, protected]

**6.29.3.16** `const std::string activemq::commands::ActiveMQDestination::TEMP _ - PREFIX` [static, protected]

**6.29.3.17** `const std::string activemq::commands::ActiveMQDestination::TEMP _ - QUEUE _ QUALIFIED _ PREFIX` [static, protected]

**6.29.3.18** `const std::string activemq::commands::ActiveMQDestination::TEMP _ - TOPIC _ QUALIFIED _ PREFIX` [static, protected]

**6.29.3.19** `const std::string activemq::commands::ActiveMQDestination::TOPIC _ - QUALIFIED _ PREFIX` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

## 6.30 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 333).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.30.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 333). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.30.2 Constructor & Destructor Documentation

**6.30.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.30.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

## 6.30.3 Member Function Documentation

**6.30.3.1** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 610), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 639), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 696).

**6.30.3.2** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 611), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 640), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 697).

**6.30.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 582), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 611), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 640), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 697).

**6.30.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 493), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 582), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 611), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 640), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 697).

**6.30.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 494), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 583), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 612), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 641), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 698).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

## 6.31 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 337).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.31.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 337). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.31.2 Constructor & Destructor Documentation

**6.31.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.31.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

## 6.31.3 Member Function Documentation

**6.31.3.1** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 496), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 585), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 614), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 647), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 704).

**6.31.3.2** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 497), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 585), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 615), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 648), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 705).

**6.31.3.3 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 497), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 586), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 615), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 648), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 705).

**6.31.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs) throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 497), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 586), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 615), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 648), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 705).

**6.31.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker.  
***dataStructure*** - Object to be un-marshaled.  
***dataIn*** - BinaryReader that provides that data.  
***bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 498), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 587), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 649), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

## 6.32 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.341).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.32.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.341). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.32.2 Constructor & Destructor Documentation

**6.32.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.32.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

## 6.32.3 Member Function Documentation

**6.32.3.1** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 500), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 589), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 618), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 643), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 700).

**6.32.3.2** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 501), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 589), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 619), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 644), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 701).

**6.32.3.3 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException )**  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 501), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 590), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 619), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 644), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 701).

**6.32.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs) throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 501), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 590), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 619), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 644), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 701).

**6.32.3.5 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 502), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 591), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 620), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 645), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 702).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

## 6.33 activemq:wireformat::openwire::marshal:v5::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.345).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal:v5::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.33.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.345). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.33.2 Constructor & Destructor Documentation

**6.33.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.33.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

### 6.33.3 Member Function Documentation

**6.33.3.1** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 504), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 593), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 622), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 651), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 708).

**6.33.3.2** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 505), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 593), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 652), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 709).

**6.33.3.3 virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \***   
***dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )**  
**[virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker  
***dataStructure*** - Object to be marshaled  
***bs*** - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 505), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 594), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 652), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 709).

**6.33.3.4 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker  
***dataStructure*** - Object to be marshaled  
***dataOut*** - BinaryReader that provides that data sink  
***bs*** - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 505), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 594), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 652), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 709).

**6.33.3.5 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 506), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 595), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 624), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 653), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 710).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

## 6.34 activemq:wireformat::openwire::marshal:v2::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.349).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.34.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p.349). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.34.2 Constructor & Destructor Documentation

**6.34.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.34.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

## 6.34.3 Member Function Documentation

**6.34.3.1** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 508), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 597), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 626), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 655), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 716).

**6.34.3.2** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 597), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 627), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 656), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 717).

**6.34.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 598), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 627), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 656), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 717).

**6.34.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 509), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 598), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 627), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 656), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 717).

**6.34.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 510), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 599), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 628), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 657), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 718).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

## 6.35 activemq:wireformat::openwire::marshal:v6::ActiveMQDestinationMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 353).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v6::ActiveMQDestinationMarshaller:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.35.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 353). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.35.2 Constructor & Destructor Documentation

**6.35.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

**6.35.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

## 6.35.3 Member Function Documentation

**6.35.3.1** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 512), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 601), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 630), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 659), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 712).

**6.35.3.2** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 513), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 601), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 631), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 660), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 713).

**6.35.3.3 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException )**  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 513), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 631), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 660), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 713).

**6.35.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs) throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 513), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 631), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 660), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 713).

**6.35.3.5 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 514), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 603), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 632), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 661), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 714).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h`

## 6.36 activemq::exceptions::ActiveMQException Class Reference

#include <src/main/activemq/exceptions/ActiveMQException.h> Inheritance diagram for activemq::exceptions::ActiveMQException:

### Public Member Functions

- **ActiveMQException** () throw ()  
*Default Constructor.*
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** \* **clone** () const  
*Clones this exception.*
- virtual **cms::CMSException** **convertToCMSException** () const  
*Converts this exception to a new CMSException.*

### 6.36.1 Constructor & Destructor Documentation

#### 6.36.1.1 activemq::exceptions::ActiveMQException::ActiveMQException () throw ()

Default Constructor.

#### 6.36.1.2 activemq::exceptions::ActiveMQException::ActiveMQException (const ActiveMQException & ex) throw ()

Copy Constructor.

#### Parameters:

**ex** The Exception whose internal data is copied into this instance.

### 6.36.1.3 `activemq::exceptions::ActiveMQException::ActiveMQException (const decaf::lang::Exception & ex) throw ()`

Copy Constructor.

#### Parameters:

*ex* The Exception whose internal data is copied into this instance.

### 6.36.1.4 `activemq::exceptions::ActiveMQException::ActiveMQException (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters:

*file* The file name where exception occurs.

*lineNumber* The line number where the exception occurred.

*msg* The message to report.

... The list of primitives that are formatted into the message.

### 6.36.1.5 `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException () throw () [virtual]`

## 6.36.2 Member Function Documentation

### 6.36.2.1 `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone () const [virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns:

Copy of this Exception object

Reimplemented from `decaf::lang::Exception` (p. 1834).

Reimplemented in `activemq::exceptions::BrokerException` (p. 867).

### 6.36.2.2 `virtual cms::CMSEException activemq::exceptions::ActiveMQException::convertToCMSEException () const [virtual]`

Converts this exception to a new CMSEException.

#### Returns:

a CMSEException with the data from this exception

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/ActiveMQException.h`

## 6.37 activemq::commands::ActiveMQMapMessage Class Reference

#include <src/main/activemq/commands/ActiveMQMapMessage.h> Inheritance diagram for activemq::commands::ActiveMQMapMessage:

### Public Member Functions

- **ActiveMQMapMessage** ()
- virtual **~ActiveMQMapMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual bool **isMarshalAware** () const  
*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*
- virtual **ActiveMQMapMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat) throw ( decaf::io::IOException )  
*Perform any processing needed before an marshal.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSEException )  
*Clears out the body of the message.*
- virtual **cms::MapMessage \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::vector< std::string > **getMapNames** () const throw ( cms::CMSEException )  
*Returns an Enumeration of all the names in the MapMessage object.*
- virtual bool **itemExists** (const std::string &name) const throw ( cms::CMSEException )  
*Indicates whether an item exists in this MapMessage object.*

- virtual bool **getBoolean** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const std::string &name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a boolean value with the specified name into the Map.*
- virtual unsigned char **getByte** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Char value of the Specified name.*
- virtual void **setChar** (const std::string &name, char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Char value with the specified name into the Map.*
- virtual double **getDouble** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Double value of the Specified name.*
- virtual void **setDouble** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Double value with the specified name into the Map.*
- virtual float **getFloat** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Float value of the Specified name.*
- virtual void **setFloat** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Float value with the specified name into the Map.*
- virtual int **getInt** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Int value of the Specified name.*

- virtual void **setInt** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Int value with the specified name into the Map.*
- virtual long long **getLong** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Long value of the Specified name.*
- virtual void **setLong** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Long value with the specified name into the Map.*
- virtual short **getShort** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Short value of the Specified name.*
- virtual void **setShort** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Short value with the specified name into the Map.*
- virtual std::string **getString** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the String value of the Specified name.*
- virtual void **setString** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a String value with the specified name into the Map.*

## Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQMAPMESSAGE** = 25

## Protected Member Functions

- **util::PrimitiveMap & getMap** () throw ( decaf::lang::exceptions::NullPointerException )  
*Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.*
- const **util::PrimitiveMap & getMap** () const throw ( decaf::lang::exceptions::NullPointerException )
- virtual void **checkMapIsUnmarshalled** () const throw ( decaf::lang::exceptions::NullPointerException )  
*Performs the unmarshal on the Map if needed, otherwise just returns.*



### 6.37.1 Constructor & Destructor Documentation

**6.37.1.1** `activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()`

**6.37.1.2** `virtual  
activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ()  
[virtual]`

### 6.37.2 Member Function Documentation

**6.37.2.1** `virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal  
(wireformat::WireFormat * wireFormat) throw ( decaf::io::IOException )  
[virtual]`

Perform any processing needed before an marshal.

#### Parameters:

*wireFormat* - the OpenWireFormat object in use.

Implements `activemq::wireformat::MarshalAware` (p. 2485).

**6.37.2.2** `virtual void ac-  
tivemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled ()  
const throw ( decaf::lang::exceptions::NullPointerException ) [protected,  
virtual]`

Performs the unmarshal on the Map if needed, otherwise just returns.

**6.37.2.3** `virtual void activemq::commands::ActiveMQMapMessage::clearBody ()  
throw ( cms::CMSException ) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 428).

**6.37.2.4** `virtual cms::MapMessage* ac-  
tivemq::commands::ActiveMQMapMessage::clone () const  
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns:

new copy of this message

Implements `cms::Message` (p. 2539).

**6.37.2.5** `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2520).

**6.37.2.6** `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2521).

**6.37.2.7** `virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 429).

**6.37.2.8** `virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Boolean value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2474).

**6.37.2.9** virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Byte value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements cms::MapMessage (p. 2475).

**6.37.2.10** virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Bytes value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements cms::MapMessage (p. 2475).

**6.37.2.11** virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Char value of the Specified name.

**Parameters:**

*name* name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements cms::MapMessage (p. 2475).

**6.37.2.12** `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Message** (p. 2523).

**6.37.2.13** `virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Double value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2476).

**6.37.2.14** `virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Float value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2476).

**6.37.2.15** `virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Int value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2476).

**6.37.2.16** `virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSEException )` [virtual]

Returns the Long value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2477).

**6.37.2.17** `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const throw ( decaf::lang::exceptions::NullPointerException )` [protected]

**6.37.2.18** `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () throw ( decaf::lang::exceptions::NullPointerException )` [protected]

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

**Returns:**

reference to a PrimitiveMap.

**6.37.2.19** `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames () const throw ( cms::CMSEException )` [virtual]

Returns an Enumeration of all the names in the MapMessage object.

**Returns:**

STL Vector of String values, each of which is the name of an item in the MapMessage

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 2477).

**6.37.2.20** `virtual short activemq::commands::ActiveMQMapMessage::getShort  
(const std::string & name) const throw ( cms::MessageFormatException,  
cms::CMSException ) [virtual]`

Returns the Short value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2477).

**6.37.2.21** `virtual std::string ac-  
tivemq::commands::ActiveMQMapMessage::getString  
(const std::string & name) const throw ( cms::MessageFormatException,  
cms::CMSException ) [virtual]`

Returns the String value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageFormatException* - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2477).

**6.37.2.22** `virtual bool ac-  
tivemq::commands::ActiveMQMapMessage::isMarshalAware () const  
[inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

**Returns:**

true if aware of marshaling

Reimplemented from `activemq::commands::Message` (p. 2526).

**6.37.2.23** `virtual bool activemq::commands::ActiveMQMapMessage::itemExists  
(const std::string & name) const throw ( cms::CMSException )  
[virtual]`

Indicates whether an item exists in this MapMessage object.

**Parameters:**

*name* String name of the Object in question

**Returns:**

boolean value indicating if the name is in the map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 2478).

**6.37.2.24** `virtual void activemq::commands::ActiveMQMapMessage::setBoolean  
(const std::string & name, bool value) throw (  
cms::MessageNotWriteableException, cms::CMSEException )` [virtual]

Sets a boolean value with the specified name into the Map.

**Parameters:**

*name* the name of the boolean

*value* the boolean value to set in the Map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the **Message** (p. 2516) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2478).

**6.37.2.25** `virtual void activemq::commands::ActiveMQMapMessage::setByte  
(const std::string & name, unsigned char value) throw (  
cms::MessageNotWriteableException, cms::CMSEException )` [virtual]

Sets a Byte value with the specified name into the Map.

**Parameters:**

*name* the name of the Byte

*value* the Byte value to set in the Map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the **Message** (p. 2516) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2478).

**6.37.2.26** `virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Bytes value with the specified name into the Map.

**Parameters:**

*name* The name of the Bytes

*value* The Bytes value to set in the Map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2479).

**6.37.2.27** `virtual void activemq::commands::ActiveMQMapMessage::setChar (const std::string & name, char value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Char value with the specified name into the Map.

**Parameters:**

*name* the name of the Char

*value* the Char value to set in the Map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2479).

**6.37.2.28** `virtual void activemq::commands::ActiveMQMapMessage::setDouble (const std::string & name, double value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Double value with the specified name into the Map.

**Parameters:**

*name* The name of the Double

*value* The Double value to set in the Map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2480).



**6.37.2.29** `virtual void activemq::commands::ActiveMQMapMessage::setFloat (const std::string & name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [virtual]`

Sets a Float value with the specified name into the Map.

**Parameters:**

*name* The name of the Float

*value* The Float value to set in the Map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2480).

**6.37.2.30** `virtual void activemq::commands::ActiveMQMapMessage::setInt (const std::string & name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [virtual]`

Sets a Int value with the specified name into the Map.

**Parameters:**

*name* The name of the Int

*value* The Int value to set in the Map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2480).

**6.37.2.31** `virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [virtual]`

Sets a Long value with the specified name into the Map.

**Parameters:**

*name* The name of the Long

*value* The Long value to set in the Map

**Exceptions:**

*CMSEException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the Message (p. 2516) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2481).

**6.37.2.32** `virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & name, short value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Sets a Short value with the specified name into the Map.

**Parameters:**

*name* The name of the Short

*value* The Short value to set in the Map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the **Message** (p. 2516) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2481).

**6.37.2.33** `virtual void activemq::commands::ActiveMQMapMessage::setString (const std::string & name, const std::string & value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Sets a String value with the specified name into the Map.

**Parameters:**

*name* The name of the String

*value* The String value to set in the Map

**Exceptions:**

*CMSException* - if the operation fails due to an internal error.

*MessageNotWriteableException* - if the **Message** (p. 2516) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2481).

**6.37.2.34** `virtual std::string activemq::commands::ActiveMQMapMessage::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2530).

## 6.37.3 Field Documentation

**6.37.3.1** `const unsigned char activemq::commands::ActiveMQMapMessage::ID_ - ACTIVEMQMAPMESSAGE = 25` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMapMessage.h`

## 6.38 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.374).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.38.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.374). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.38.2 Constructor & Destructor Documentation

**6.38.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

**6.38.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

## 6.38.3 Member Function Documentation

**6.38.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.38.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.38.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.38.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.38.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700).

**6.38.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

**6.38.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h`

## 6.39 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.378).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.39.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.378). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.39.2 Constructor & Destructor Documentation

**6.39.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

**6.39.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

## 6.39.3 Member Function Documentation

**6.39.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject(const std::string&)` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.39.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType()` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.39.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.39.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.39.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2715).

**6.39.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

**6.39.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h`

## 6.40 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.382).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.40.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.382). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.40.2 Constructor & Destructor Documentation

**6.40.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` `[inline]`

**6.40.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` `[inline, virtual]`

## 6.40.3 Member Function Documentation

**6.40.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.40.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.40.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.40.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.40.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2710).

**6.40.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

**6.40.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h`

## 6.41 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.386).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.41.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.386). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.41.2 Constructor & Destructor Documentation

6.41.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()** [inline]

6.41.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()** [inline, virtual]

## 6.41.3 Member Function Documentation

6.41.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const** [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1611).

6.41.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType() const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1617).

6.41.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure\* dataStructure, decaf::io::DataOutputStream\* dataOut) throw (decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.41.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.41.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2695).

**6.41.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

**6.41.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h`

## 6.42 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.390).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.42.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.390). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.42.2 Constructor & Destructor Documentation

**6.42.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

**6.42.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

## 6.42.3 Member Function Documentation

**6.42.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.42.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.42.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.42.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.42.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705).

**6.42.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

**6.42.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h`

## 6.43 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.394).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.43.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.394). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.43.2 Constructor & Destructor Documentation

**6.43.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

**6.43.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

## 6.43.3 Member Function Documentation

**6.43.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.43.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.43.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.43.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.43.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720).

**6.43.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

**6.43.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h`

## 6.44 activemq::commands::ActiveMQMessage Class Reference

#include <src/main/activemq/commands/ActiveMQMessage.h> Inheritance diagram for activemq::commands::ActiveMQMessage:

### Public Member Functions

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual **ActiveMQMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQMESSAGE** = 23

#### 6.44.1 Constructor & Destructor Documentation

- 6.44.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage** ()
- 6.44.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage** ()  
[inline, virtual]

#### 6.44.2 Member Function Documentation

- 6.44.2.1 **virtual cms::Message\* activemq::commands::ActiveMQMessage::clone** ()  
const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

**Returns:**

new copy of this message

Implements **cms::Message** (p. 2539).

**6.44.2.2** `virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2520).

**6.44.2.3** `virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2521).

**6.44.2.4** `virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 429).

**6.44.2.5** `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Message** (p. 2523).

#### 6.44.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ()` `const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

##### Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2530).

### 6.44.3 Field Documentation

#### 6.44.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_-` `ACTIVEMQMESSAGE = 23` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`

## 6.45 activemq:wireformat::openwire::marshal:v3::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal:v3::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.45.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.45.2 Constructor & Destructor Documentation

**6.45.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.45.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.45.3 Member Function Documentation

**6.45.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.45.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.45.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.45.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.45.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700).

**6.45.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

**6.45.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h`

## 6.46 activemq:wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 405).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.46.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 405).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.46.2 Constructor & Destructor Documentation

**6.46.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.46.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.46.3 Member Function Documentation

**6.46.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.46.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.46.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.46.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.46.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2715).

**6.46.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

**6.46.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h`

## 6.47 activemq:wireformat::openwire::marshal:v4::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 409).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal:v4::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.47.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 409).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.47.2 Constructor & Destructor Documentation

**6.47.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.47.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.47.3 Member Function Documentation

**6.47.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.47.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.47.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.47.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.47.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2710).

**6.47.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

**6.47.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h`

## 6.48 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 413).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.48.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 413).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.48.2 Constructor & Destructor Documentation

**6.48.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.48.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.48.3 Member Function Documentation

**6.48.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.48.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.48.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.48.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.48.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2695).

**6.48.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

**6.48.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h`

## 6.49 activemq:wireformat::openwire::marshal:v2::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 417).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.49.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 417).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.49.2 Constructor & Destructor Documentation

**6.49.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.49.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.49.3 Member Function Documentation

**6.49.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.49.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.49.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.49.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.49.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705).

**6.49.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

**6.49.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h`

## 6.50 activemq:wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 421).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.50.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQMessageMarshaller** (p. 421).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.50.2 Constructor & Destructor Documentation

**6.50.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

**6.50.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

## 6.50.3 Member Function Documentation

**6.50.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.50.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.50.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.50.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.50.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720).

**6.50.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

**6.50.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h`

## 6.51 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

#include <src/main/activemq/commands/ActiveMQMessageTemplate.h> Inheritance diagram for activemq::commands::ActiveMQMessageTemplate< T >:

### Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual **~ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw ( cms::IllegalStateException, cms::CMSEException )  
*Acknowledges all consumed messages of the session of this consumed message.*
- virtual void **onSend** ()  
*Resets the **Message** (p. 2516) to a Read-Only **state** (p. 95).*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSEException )  
*Clears out the body of the message.*
- virtual void **clearProperties** () throw ( cms::CMSEException )  
*Clears the message properties.*
- virtual std::vector< std::string > **getPropertyNames** () const throw ( cms::CMSEException )  
*Retrieves the property names.*
- virtual bool **propertyExists** (const std::string &name) const throw ( cms::CMSEException )  
*Indicates whether or not a given property exists.*
- virtual bool **getBooleanProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a float property.*

- virtual int **getIntProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a int property.*

- virtual long long **getLongProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a long property.*

- virtual short **getShortProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a short property.*

- virtual std::string **getStringProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a string property.*

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a boolean property.*

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a byte property.*

- virtual void **setDoubleProperty** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a double property.*

- virtual void **setFloatProperty** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a float property.*

- virtual void **setIntProperty** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a int property.*

- virtual void **setLongProperty** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a short property.*

- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a string property.*

- virtual std::string **getCMSCorrelationID** () const throw ( cms::CMSEException )



*Get the Correlation Id for this message.*

- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw ( cms::CMSException )

*Sets the Correlation Id used by this message.*

- virtual int **getCMSDeliveryMode** () const throw ( cms::CMSException )

*Gets the DeliveryMode for this message.*

- virtual void **setCMSDeliveryMode** (int mode) throw ( cms::CMSException )

*Sets the DeliveryMode for this message.*

- virtual const cms::Destination \* **getCMSDestination** () const throw ( cms::CMSException )

*Gets the Destination for this **Message** (p. 2516), returns a.*

- virtual void **setCMSDestination** (const cms::Destination \*destination) throw ( cms::CMSException )

*Sets the Destination for this message.*

- virtual long long **getCMSExpiration** () const throw ( cms::CMSException )

*Gets the Expiration Time for this **Message** (p. 2516).*

- virtual void **setCMSExpiration** (long long expireTime) throw ( cms::CMSException )

*Sets the Expiration Time for this message.*

- virtual std::string **getCMSMessageID** () const throw ( cms::CMSException )

*Gets the CMS **Message** (p. 2516) Id for this **Message** (p. 2516).*

- virtual void **setCMSMessageID** (const std::string &id AMQCPP\_UNUSED) throw ( cms::CMSException )

*Sets the CMS **Message** (p. 2516) Id for this message.*

- virtual int **getCMSPriority** () const throw ( cms::CMSException )

*Gets the Priority Value for this **Message** (p. 2516).*

- virtual void **setCMSPriority** (int priority) throw ( cms::CMSException )

*Sets the Priority Value for this message.*

- virtual bool **getCMSRedelivered** () const throw ( cms::CMSException )

*Gets the Redelivered Flag for this **Message** (p. 2516).*

- virtual void **setCMSRedelivered** (bool redelivered AMQCPP\_UNUSED) throw ( cms::CMSException )

*Sets the Redelivered Flag for this message.*

- virtual const cms::Destination \* **getCMSReplyTo** () const throw ( cms::CMSException )

*Gets the CMS Reply To Address for this **Message** (p. 2516).*

- virtual void **setCMSReplyTo** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*Sets the CMS Reply To Address for this message.*
- virtual long long **getCMSTimestamp** () const throw ( cms::CMSEException )  
*Gets the Time Stamp for this **Message** (p. 2516).*
- virtual void **setCMSTimestamp** (long long timeStamp) throw ( cms::CMSEException )  
*Sets the Time Stamp for this message.*
- virtual std::string **getCMSType** () const throw ( cms::CMSEException )  
*Gets the CMS **Message** (p. 2516) Type for this **Message** (p. 2516).*
- virtual void **setCMSType** (const std::string &type) throw ( cms::CMSEException )  
*Sets the CMS **Message** (p. 2516) Type for this message.*

## Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T
>
```

### 6.51.1 Constructor & Destructor Documentation

- 6.51.1.1 `template<typename T>  
activemq::commands::ActiveMQMessageTemplate< T  
>::ActiveMQMessageTemplate () [inline]`
- 6.51.1.2 `template<typename T> virtual  
activemq::commands::ActiveMQMessageTemplate< T  
>::~~ActiveMQMessageTemplate () [inline, virtual]`

### 6.51.2 Member Function Documentation

- 6.51.2.1 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::acknowledge () const throw ( cms::IllegalStateException,  
cms::CMSEException ) [inline, virtual]`

Acknowledges all consumed messages of the session of this consumed message.

- 6.51.2.2 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::clearBody () throw ( cms::CMSEException ) [inline, virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 236), **activemq::commands::ActiveMQMapMessage** (p. 363), **activemq::commands::ActiveMQStreamMessage** (p. 540), and **activemq::commands::ActiveMQTextMessage** (p. 663).

**6.51.2.3** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::clearProperties () throw ( cms::CMSEException ) [inline, virtual]`

Clears the message properties. Does not clear the body or header values.

**6.51.2.4** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 2521).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 206), **activemq::commands::ActiveMQBytesMessage** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 541), and **activemq::commands::ActiveMQTextMessage** (p. 664).

**6.51.2.5** `template<typename T> void  
activemq::commands::ActiveMQMessageTemplate< T  
>::failIfReadOnlyBody () const [inline, protected]`

**6.51.2.6** `template<typename T> void  
activemq::commands::ActiveMQMessageTemplate< T  
>::failIfReadOnlyProperties () const [inline, protected]`

**6.51.2.7** `template<typename T> void  
activemq::commands::ActiveMQMessageTemplate< T  
>::failIfWriteOnlyBody () const [inline, protected]`

**6.51.2.8** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::getBooleanProperty (const std::string & name) const throw (  
cms::MessageFormatException, cms::CMSEException ) [inline, virtual]`

Gets a boolean property.

#### Parameters:

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

```
6.51.2.9  template<typename T> virtual unsigned char
          activemq::commands::ActiveMQMessageTemplate< T >::getBytesProperty
          (const std::string & name) const throw ( cms::MessageFormatException,
          cms::CMSEException ) [inline, virtual]
```

Gets a byte property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

```
6.51.2.10 template<typename T> virtual std::string
          activemq::commands::ActiveMQMessageTemplate< T
          >::getCMSCorrelationID () const throw ( cms::CMSEException )
          [inline, virtual]
```

Get the Correlation Id for this message.

**Returns:**

string representation of the correlation Id

**Exceptions:**

*CMSEException*

```
6.51.2.11 template<typename T> virtual int
          activemq::commands::ActiveMQMessageTemplate< T
          >::getCMSDeliveryMode () const throw ( cms::CMSEException )
          [inline, virtual]
```

Gets the DeliveryMode for this message.

**Returns:**

DeliveryMode enumerated value.

Exceptions:

*CMSException*

**6.51.2.12** `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSDestination () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Destination for this **Message** (p. 2516), returns a.

Returns:

Destination object

Exceptions:

*CMSException*

**6.51.2.13** `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSExpiration () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Expiration Time for this **Message** (p. 2516).

Returns:

time value

Exceptions:

*CMSException*

**6.51.2.14** `template<typename T> virtual std::string  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSMessageID () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the CMS **Message** (p. 2516) Id for this **Message** (p. 2516).

Returns:

time value

Exceptions:

*CMSException*

**6.51.2.15** `template<typename T> virtual int  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSPriority () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Priority Value for this **Message** (p. 2516).

**Returns:**

priority value

**Exceptions:**

*CMSException*

**6.51.2.16** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSRedelivered () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Redelivered Flag for this **Message** (p. 2516).

**Returns:**

redelivered value

**Exceptions:**

*CMSException*

**6.51.2.17** `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSReplyTo () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the CMS Reply To Address for this **Message** (p. 2516).

**Returns:**

Reply To Value

**Exceptions:**

*CMSException*

**6.51.2.18** `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSTimestamp () const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Time Stamp for this **Message** (p. 2516).

**Returns:**

time stamp value

**Exceptions:**

*CMSEException*

**6.51.2.19**    `template<typename T> virtual std::string  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSType () const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the CMS Message (p. 2516) Type for this Message (p. 2516).

**Returns:**

type value

**Exceptions:**

*CMSEException*

**6.51.2.20**    `template<typename T> virtual double  
activemq::commands::ActiveMQMessageTemplate< T  
>::getDoubleProperty (const std::string & name) const throw (  
cms::MessageFormatException, cms::CMSEException ) [inline, virtual]`

Gets a double property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

**6.51.2.21**    `template<typename T> virtual float  
activemq::commands::ActiveMQMessageTemplate< T  
>::getFloatProperty (const std::string & name) const throw (  
cms::MessageFormatException, cms::CMSEException ) [inline, virtual]`

Gets a float property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

```
6.51.2.22  template<typename T> virtual int
             activemq::commands::ActiveMQMessageTemplate< T
             >::getIntProperty (const std::string & name) const throw (
             cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a int property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

```
6.51.2.23  template<typename T> virtual long long
             activemq::commands::ActiveMQMessageTemplate< T
             >::getLongProperty (const std::string & name) const throw (
             cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a long property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.



```
6.51.2.24  template<typename T> virtual std::vector<std::string>
           activemq::commands::ActiveMQMessageTemplate< T
           >::getPropertyNames () const throw ( cms::CMSException ) [inline,
           virtual]
```

Retrieves the property names.

**Returns:**

The complete set of property names currently in this message.

```
6.51.2.25  template<typename T> virtual short
           activemq::commands::ActiveMQMessageTemplate< T
           >::getShortProperty (const std::string & name) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a short property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

```
6.51.2.26  template<typename T> virtual std::string
           activemq::commands::ActiveMQMessageTemplate< T
           >::getStringProperty (const std::string & name) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a string property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSException* if the property does not exist.

*MessageFormatException* - if this type conversion is invalid.

**6.51.2.27** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::onSend () [inline, virtual]`

Resets the `Message` (p. 2516) to a Read-Only state (p. 95).

Reimplemented from `activemq::commands::Message` (p. 2527).

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 238), and `activemq::commands::ActiveMQStreamMessage` (p. 541).

**6.51.2.28** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::propertyExists (const std::string & name) const throw ( cms::CMSException ) [inline, virtual]`

Indicates whether or not a given property exists.

**Parameters:**

*name* The name of the property to look up.

**Returns:**

True if the property exists in this message.

**6.51.2.29** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setBooleanProperty (const std::string & name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [inline, virtual]`

Sets a boolean property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

**6.51.2.30** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setByteProperty (const std::string & name, unsigned char value)  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[inline, virtual]`

Sets a byte property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

```
6.51.2.31  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSCorrelationID (const std::string & correlationId) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the Correlation Id used by this message.

**Parameters:**

*correlationId* - String representing the correlation id.

**Exceptions:**

*CMSException*

```
6.51.2.32  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSDeliveryMode (int mode) throw ( cms::CMSException )
            [inline, virtual]
```

Sets the DeliveryMode for this message.

**Parameters:**

*mode* - DeliveryMode enumerated value.

**Exceptions:**

*CMSException*

```
6.51.2.33  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSDestination (const cms::Destination * destination) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the Destination for this message.

**Parameters:**

*destination* - Destination Object

**Exceptions:**

*CMSException*

**6.51.2.34** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSExpiration (long long expireTime) throw ( cms::CMSException )` [inline, virtual]

Sets the Expiration Time for this message.

**Parameters:**

*expireTime* - time value

**Exceptions:**

*CMSException*

**6.51.2.35** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSMessageID (const std::string &id AMQCPP_UNUSED)  
throw ( cms::CMSException )` [inline, virtual]

Sets the CMS Message (p. 2516) Id for this message.

**Parameters:**

*id* - time value

**Exceptions:**

*CMSException*

**6.51.2.36** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSPriority (int priority) throw ( cms::CMSException )` [inline, virtual]

Sets the Priority Value for this message.

**Parameters:**

*priority* - priority value for this message

**Exceptions:**

*CMSException*

**6.51.2.37** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSRedelivered (bool redelivered AMQCPP_UNUSED) throw ( cms::CMSException )` [inline, virtual]

Sets the Redelivered Flag for this message.

Parameters:

*redelivered* - boolean redelivered value

Exceptions:

*CMSException*

```
6.51.2.38 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSReplyTo (const cms::Destination * destination) throw (
cms::CMSException ) [inline, virtual]
```

Sets the CMS Reply To Address for this message.

Parameters:

*destination* Pointer to the CMS Destination that is the Reply-To value.

Exceptions:

*CMSException*

```
6.51.2.39 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSTimestamp (long long timeStamp) throw (
cms::CMSException ) [inline, virtual]
```

Sets the Time Stamp for this message.

Parameters:

*timeStamp* - integer time stamp value

Exceptions:

*CMSException*

```
6.51.2.40 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSType (const std::string & type) throw ( cms::CMSException )
[inline, virtual]
```

Sets the CMS **Message** (p. 2516) Type for this message.

Parameters:

*type* - message type value string

Exceptions:

*CMSException*

**6.51.2.41** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setDoubleProperty (const std::string & name, double value) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [inline,  
virtual]`

Sets a double property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

**6.51.2.42** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setFloatProperty (const std::string & name, float value) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [inline,  
virtual]`

Sets a float property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

**6.51.2.43** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setIntProperty (const std::string & name, int value) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [inline,  
virtual]`

Sets a int property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

```
6.51.2.44  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setLongProperty (const std::string & name, long long value) throw (
           cms::MessageNotWriteableException, cms::CMSException ) [inline,
           virtual]
```

Sets a long property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

```
6.51.2.45  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setShortProperty (const std::string & name, short value) throw (
           cms::MessageNotWriteableException, cms::CMSException ) [inline,
           virtual]
```

Sets a short property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

*MessageNotWriteableException* - if properties are read-only

```
6.51.2.46  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setStringProperty (const std::string & name, const std::string &
           value) throw ( cms::MessageNotWriteableException, cms::CMSException
           ) [inline, virtual]
```

Sets a string property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* - if the name is an empty string.

***MessageNotWriteableException*** - if properties are read-only

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessageTemplate.h`



## 6.52 activemq::commands::ActiveMQObjectMessage Class Reference

#include <src/main/activemq/commands/ActiveMQObjectMessage.h> Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

### Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQObjectMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQOBJECTMESSAGE** = 26

## 6.52.1 Constructor & Destructor Documentation

**6.52.1.1** `activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage()`

**6.52.1.2** `virtual activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage()` [inline, virtual]

## 6.52.2 Member Function Documentation

**6.52.2.1** `virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone()` const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns:

new copy of this message

Implements `cms::Message` (p. 2539).

**6.52.2.2** `virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure()` const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2520).

**6.52.2.3** `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2521).

**6.52.2.4** `virtual bool activemq::commands::ActiveMQObjectMessage::equals(const DataStructure * value)` const [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 429).

**6.52.2.5** `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from `activemq::commands::Message` (p. 2523).

**6.52.2.6** `virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2530).

### 6.52.3 Field Documentation

**6.52.3.1** `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ACTIVEMQOBJECTMESSAGE = 26 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQObjectMessage.h`

## 6.53 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 446).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.53.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 446). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.53.2 Constructor & Destructor Documentation

**6.53.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.53.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

### 6.53.3 Member Function Documentation

**6.53.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

#### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.53.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaller.

#### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.53.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.53.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.53.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700).

```

6.53.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

```

6.53.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h

## 6.54 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 450).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.54.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 450). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



---

6.54.2 Constructor & Destructor Documentation

---

**6.54.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.54.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

## 6.54.3 Member Function Documentation

**6.54.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

**Returns:**

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.54.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns:**

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.54.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.54.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.54.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2715).

```

6.54.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

```

6.54.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h

## 6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 454).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.55.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 454). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

---

**6.55.2 Constructor & Destructor Documentation**

**6.55.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.55.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

**6.55.3 Member Function Documentation**

**6.55.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

**Returns:**

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.55.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns:**

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.55.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.55.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.55.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2710).

```

6.55.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

```

6.55.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h

## 6.56 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 458).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.56.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 458). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.56.2 Constructor & Destructor Documentation

**6.56.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.56.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

## 6.56.3 Member Function Documentation

**6.56.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.56.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.56.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.56.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.56.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2695).

```

6.56.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

```

6.56.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h

## 6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.57.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

---

**6.57.2 Constructor & Destructor Documentation**

**6.57.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.57.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

**6.57.3 Member Function Documentation**

**6.57.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

**Returns:**

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.57.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns:**

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.57.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.57.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.57.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705).

```

6.57.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

```

6.57.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h

## 6.58 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 466).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.58.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 466). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.58.2 Constructor & Destructor Documentation

**6.58.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

**6.58.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

## 6.58.3 Member Function Documentation

**6.58.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.58.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.58.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.58.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.58.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720).

```

6.58.3.6 virtual void ac-
      tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal
      (OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataOutputStream * dataOut,
      utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

```

6.58.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightUnma
      (OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
      * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h

## 6.59 activemq::core::ActiveMQProducer Class Reference

#include <src/main/activemq/core/ActiveMQProducer.h> Inheritance diagram for activemq::core::ActiveMQProducer:

### Public Member Functions

- **ActiveMQProducer** (**ActiveMQSession** \*session, const **Pointer**<**commands::ProducerId** > &producerId, const **Pointer**<**commands::ActiveMQDestination** > &destination, long long sendTimeout)  
*Constructor, creates an instance of an **ActiveMQProducer** (p. 470).*
- virtual ~**ActiveMQProducer** ()
- virtual void **close** () throw ( cms::CMSException )  
*Closes the Consumer.*
- virtual void **send** (cms::Message \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (cms::Message \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const cms::Destination \*destination, cms::Message \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const cms::Destination \*destination, cms::Message \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSException )  
*Sets if Message Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const throw ( cms::CMSEException )  
*Gets if Message Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )  
*Gets if Message Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*
- virtual void **setSendTimeout** (long long time) throw ( cms::CMSEException )  
*Sets the Send Timeout that this Producers sends messages with.*
- virtual long long **getSendTimeout** () const throw ( cms::CMSEException )  
*Gets the Send Timeout that this producer sends messages with.*
- bool **isClosed** () const
- const **Pointer**< **commands::ProducerInfo** > & **getProducerInfo** () const  
*Retries this object ProducerInfo pointer.*
- const **Pointer**< **commands::ProducerId** > & **getProducerId** () const  
*Retries this object ProducerId or NULL if closed.*
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)  
*Handles the work of Processing a ProducerAck Command from the Broker.*

## 6.59.1 Constructor & Destructor Documentation

**6.59.1.1** **activemq::core::ActiveMQProducer::ActiveMQProducer**  
(**ActiveMQSession** \* *session*, const **Pointer**< **commands::ProducerId** >  
& *producerId*, const **Pointer**< **commands::ActiveMQDestination** > &  
*destination*, long long *sendTimeout*)

Constructor, creates an instance of an **ActiveMQProducer** (p. 470).

### Parameters:

***session*** The Session which is the parent of this Producer.

***producerId*** Pointer to a ProducerId object which identifies this producer.

***destination*** The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.

***sendTimeout*** The configured send timeout for this Producer.

**6.59.1.2** `virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer ()`  
[virtual]

## 6.59.2 Member Function Documentation

**6.59.2.1** `virtual void activemq::core::ActiveMQProducer::close () throw (`  
`cms::CMSException )` [virtual]

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

### Exceptions:

***CMSException***

Implements **cms::Closeable** (p. 1148).

**6.59.2.2** `virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const`  
`throw ( cms::CMSException )` [inline, virtual]

Gets the delivery mode for this Producer.

### Returns:

The DeliveryMode

Implements **cms::MessageProducer** (p. 2726).

**6.59.2.3** `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID ()`  
`const throw ( cms::CMSException )` [inline, virtual]

Gets if Message Ids are disabled for this Producer.

### Returns:

a boolean indicating **state** (p. 95) enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 2727).

**6.59.2.4** `virtual bool ac-`  
`tivemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const`  
`throw ( cms::CMSException )` [inline, virtual]

Gets if Message Time Stamps are disabled for this Producer.

**Returns:**

boolean indicating **state** (p. 95) of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2727).

**6.59.2.5** `virtual int activemq::core::ActiveMQProducer::getPriority () const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

**Returns:**

int based priority level

Implements **cms::MessageProducer** (p. 2727).

**6.59.2.6** `const Pointer<commands::ProducerId>& activemq::core::ActiveMQProducer::getProducerId () const [inline]`

Retries this object ProducerId or NULL if closed.

**Returns:**

ProducerId Reference

**6.59.2.7** `const Pointer<commands::ProducerInfo>& activemq::core::ActiveMQProducer::getProducerInfo () const [inline]`

Retries this object ProducerInfo pointer.

**Returns:**

ProducerInfo Reference

**6.59.2.8** `virtual long long activemq::core::ActiveMQProducer::getSendTimeout () const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

**Returns:**

The default send timeout value in milliseconds.

**6.59.2.9** `virtual long long activemq::core::ActiveMQProducer::getTimeToLive () const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

**Returns:**

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2728).

**6.59.2.10** `bool activemq::core::ActiveMQProducer::isClosed () const [inline]`**Returns:**

true if this Producer has been closed.

**6.59.2.11** `virtual void activemq::core::ActiveMQProducer::onProducerAck (const commands::ProducerAck & ack) [virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

**Parameters:**

*ack* - The ProducerAck message received from the Broker.

**6.59.2.12** `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*destination* The destination on which to send the message

*message* The message to be sent.

*deliveryMode* The delivery mode to be used.

*priority* The priority for this message.

*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

*CMSException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2728).



**6.59.2.13** virtual void activemq::core::ActiveMQProducer::send (const cms::Destination \* *destination*, cms::Message \* *message*) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

**Parameters:**

*destination* The destination on which to send the message

*message* the message to be sent.

**Exceptions:**

*CMSException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements cms::MessageProducer (p. 2728).

**6.59.2.14** virtual void activemq::core::ActiveMQProducer::send (cms::Message \* *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*message* The message to be sent.

*deliveryMode* The delivery mode to be used.

*priority* The priority for this message.

*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

*CMSException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements cms::MessageProducer (p. 2729).

**6.59.2.15** `virtual void activemq::core::ActiveMQProducer::send (cms::Message * message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

**Parameters:**

*message* The message to be sent.

**Exceptions:**

*CMSException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2729).

**6.59.2.16** `virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int mode) throw ( cms::CMSException ) [inline, virtual]`

Sets the delivery mode for this Producer.

**Parameters:**

*mode* - The DeliveryMode to use for Message sends.

Implements **cms::MessageProducer** (p. 2730).

**6.59.2.17** `virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool value) throw ( cms::CMSException ) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

**Parameters:**

*value* - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2730).

**6.59.2.18** `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool value) throw ( cms::CMSException ) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

**Parameters:**

*value* - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2730).

**6.59.2.19** `virtual void activemq::core::ActiveMQProducer::setPriority (int priority) throw ( cms::CMSException ) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

**Parameters:**

*priority* int value for Priority level

Implements **cms::MessageProducer** (p. 2731).

**6.59.2.20** `virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long time) throw ( cms::CMSException ) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

**Parameters:**

*time* The new default send timeout value in milliseconds.

**6.59.2.21** `virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long time) throw ( cms::CMSException ) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

**Parameters:**

*time* The new default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2731).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

## 6.60 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3126) object.

#include <src/main/activemq/util/ActiveMQProperties.h> Inheritance diagram for activemq::util::ActiveMQProperties:

### Public Member Functions

- **ActiveMQProperties** ()
- virtual **~ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperties** (decaf::util::Properties &props)
- virtual bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const CMSProperties \*source)  
*Copies the contents of the given properties object to this one.*
- virtual CMSProperties \* **clone** () const  
*Clones this object.*
- virtual void **clear** ()  
*Clears all properties from the map.*
- virtual std::string **toString** () const  
*Formats the contents of the Properties Object into a string that can be logged, etc.*

### 6.60.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3126) object.

**Since:**

2.0

### 6.60.2 Constructor & Destructor Documentation

**6.60.2.1** `activemq::util::ActiveMQProperties::ActiveMQProperties ()`

**6.60.2.2** `virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties ()`  
[virtual]

### 6.60.3 Member Function Documentation

**6.60.3.1** `virtual void activemq::util::ActiveMQProperties::clear ()` [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 1166).

**6.60.3.2** `virtual CMSProperties* activemq::util::ActiveMQProperties::clone ()`  
`const` [virtual]

Clones this object.

**Returns:**

a replica of this object.

Implements **cms::CMSProperties** (p. 1166).

**6.60.3.3** `virtual void activemq::util::ActiveMQProperties::copy (const CMSProperties * source)` [virtual]

Copies the contents of the given properties object to this one.

**Parameters:**

*source* The source properties object.

**6.60.3.4** `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties () const` [inline, virtual]

**6.60.3.5** `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()` [inline, virtual]

**6.60.3.6** `virtual std::string activemq::util::ActiveMQProperties::getProperty (const std::string & name, const std::string & defaultValue) const` [inline, virtual]

Looks up the value for the given property.

**Parameters:**

*name* the name of the property to be looked up.

*defaultValue* The value to be returned if the given property does not exist.

**Returns:**

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implements `cms::CMSProperties` (p. 1166).

**6.60.3.7** `virtual const char* activemq::util::ActiveMQProperties::getProperty (const std::string & name) const` [inline, virtual]

Looks up the value for the given property.

**Parameters:**

*name* The name of the property to be looked up.

**Returns:**

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements `cms::CMSProperties` (p. 1167).

**6.60.3.8** `virtual bool activemq::util::ActiveMQProperties::hasProperty (const std::string & name) const` [inline, virtual]

Check to see if the Property exists in the set.

**Parameters:**

*name* - property name to check for in this properties set.

**Returns:**

true if property exists, false otherwise.

Implements `cms::CMSProperties` (p. 1167).

**6.60.3.9 virtual bool activemq::util::ActiveMQProperties::isEmpty () const**  
[inline, virtual]

Returns true if the properties object is empty.

**Returns:**

true if empty

Implements **cms::CMSProperties** (p. 1167).

**6.60.3.10 virtual void activemq::util::ActiveMQProperties::remove (const std::string & name)** [inline, virtual]

Removes the property with the given name.

**Parameters:**

*name* the name of the property to remove.

Implements **cms::CMSProperties** (p. 1167).

**6.60.3.11 virtual void activemq::util::ActiveMQProperties::setProperties (decaf::util::Properties & props)** [inline, virtual]**6.60.3.12 virtual void activemq::util::ActiveMQProperties::setProperty (const std::string & name, const std::string & value)** [inline, virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

**Parameters:**

*name* The name of the value to be written.

*value* The value to be written.

Implements **cms::CMSProperties** (p. 1168).

**6.60.3.13 virtual std::vector< std::pair< std::string, std::string > > activemq::util::ActiveMQProperties::toArray () const** [inline, virtual]

Method that serializes the contents of the property map to an array.

**Returns:**

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 1168).

**6.60.3.14 virtual std::string activemq::util::ActiveMQProperties::toString () const**  
[inline, virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

**Returns:**

string value of this object.

Implements **cms::CMSProperties** (p. 1168).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`



## 6.61 activemq::commands::ActiveMQQueue Class Reference

#include <src/main/activemq/commands/ActiveMQQueue.h> Inheritance diagram for activemq::commands::ActiveMQQueue:

### Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual ~**ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **ActiveMQQueue** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination \* **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual cms::Destination \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const cms::Destination &source)  
*Copies the contents of the given Destinastion object to this one.*
- virtual const cms::CMSProperties & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getQueueName** () const throw ( cms::CMSException )  
*Gets the name of this queue.*

### Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQQUEUE** = 100

### 6.61.1 Constructor & Destructor Documentation

**6.61.1.1** `activemq::commands::ActiveMQQueue::ActiveMQQueue ()`

**6.61.1.2** `activemq::commands::ActiveMQQueue::ActiveMQQueue (const std::string & name)`

**6.61.1.3** `virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue ()`  
[inline, virtual]

### 6.61.2 Member Function Documentation

**6.61.2.1** `virtual cms::Destination* activemq::commands::ActiveMQQueue::clone ()`  
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

**Returns:**

cloned copy of this object

Implements **cms::Destination** (p.1724).

**6.61.2.2** `virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p.324).

**6.61.2.3** `virtual void activemq::commands::ActiveMQQueue::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

**Parameters:**

*source* The source Destination object.

**6.61.2.4** `virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p.325).

**6.61.2.5 virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure \* *value*) const [virtual]**

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

**6.61.2.6 virtual const cms::Destination\* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]****Returns:**

the **cms::Destination** (p. 1723) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

**6.61.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]**

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns:**

const reference to a properties object.

Implements **cms::Destination** (p. 1724).

**6.61.2.8 virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const [virtual]**

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

**Returns:**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

**6.61.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const [inline, virtual]**

Retrieve the Destination Type for this Destination.

**Returns:**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 327).

References **cms::Destination::QUEUE**.

**6.61.2.10** `virtual std::string activemq::commands::ActiveMQQueue::getQueueName  
() const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this queue.

**Returns:**

The queue name.

Implements **cms::Queue** (p. 3148).

**6.61.2.11** `virtual std::string activemq::commands::ActiveMQQueue::toString ()  
const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 330).

### 6.61.3 Field Documentation

**6.61.3.1** `const unsigned char activemq::commands::ActiveMQQueue::ID_-  
ACTIVEMQQUEUE = 100 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

## 6.62 activemq::core::ActiveMQQueueBrowser Class Reference

#include <src/main/activemq/core/ActiveMQQueueBrowser.h> Inheritance diagram for activemq::core::ActiveMQQueueBrowser:

### Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** \*session, const **Pointer**<**commands::ConsumerId** > &consumerId, const **Pointer**<**commands::ActiveMQDestination** > &destination, const std::string &selector, bool dispatchAsync)
- virtual ~**ActiveMQQueueBrowser** ()
- virtual const **cms::Queue** \* **getQueue** () const throw ( cms::CMSException )
- virtual std::string **getMessageSelector** () const throw ( cms::CMSException )
- virtual **cms::MessageEnumeration** \* **getEnumeration** () throw ( cms::CMSException )

*Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.*

- virtual void **close** () throw ( cms::CMSException )

*Closes this object and deallocates the appropriate resources.*

- virtual bool **hasMoreMessages** ()

*Returns true if there are more Message in the Browser that can be retrieved via the **nextMessage** method.*

- virtual **cms::Message** \* **nextMessage** () throw ( cms::CMSException )

*Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.*

### Friends

- class **Browser**

## 6.62.1 Constructor & Destructor Documentation

- 6.62.1.1** `activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser (ActiveMQSession * session, const Pointer< commands::ConsumerId > & consumerId, const Pointer< commands::ActiveMQDestination > & destination, const std::string & selector, bool dispatchAsync)`
- 6.62.1.2** `virtual  
activemq::core::ActiveMQQueueBrowser::~~ActiveMQQueueBrowser ()  
[virtual]`

## 6.62.2 Member Function Documentation

- 6.62.2.1** `virtual void activemq::core::ActiveMQQueueBrowser::close () throw ( cms::CMSEException ) [virtual]`

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

### Exceptions:

*CMSEException* - If an error occurs while the resource is being closed.

Implements `cms::Closeable` (p.1148).

- 6.62.2.2** `virtual cms::MessageEnumeration* activemq::core::ActiveMQQueueBrowser::getEnumeration ()  
throw ( cms::CMSEException ) [virtual]`

Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them. The pointer returned is owned by the browser and should not be deleted by the client application.

### Returns:

a pointer to a Queue Enumeration, this Pointer is owned by the QueueBrowser and should not be deleted by the client.

### Exceptions:

*CMSEException* if an internal error occurs.

Implements `cms::QueueBrowser` (p.3153).

- 6.62.2.3** `virtual std::string activemq::core::ActiveMQQueueBrowser::getMessageSelector () const throw ( cms::CMSEException ) [virtual]`

### Returns:

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

**Exceptions:**

*CMSEException* if an internal error occurs.

Implements **cms::QueueBrowser** (p. 3154).

```
6.62.2.4 virtual const cms::Queue* ac-
        tivemq::core::ActiveMQQueueBrowser::getQueue () const
        throw ( cms::CMSEException ) [virtual]
```

**Returns:**

the Queue that this browser is listening on.

**Exceptions:**

*CMSEException* if an internal error occurs.

Implements **cms::QueueBrowser** (p. 3154).

```
6.62.2.5 virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages ()
        [virtual]
```

Returns true if there are more Message in the Browser that can be retrieved via the **nextMessage** method. If this method returns false and the **nextMessage** method is called then an Exception will be thrown.

**Returns:**

true if more Message's are available in the Browser.

Implements **cms::MessageEnumeration** (p. 2659).

```
6.62.2.6 virtual cms::Message* ac-
        tivemq::core::ActiveMQQueueBrowser::nextMessage ()
        throw ( cms::CMSEException ) [virtual]
```

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown. If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

**Returns:**

The next Message in the Queue.

**Exceptions:**

*CMSEException* if no more Message's currently in the Queue.

Implements **cms::MessageEnumeration** (p. 2660).

## 6.62.3 Friends And Related Function Documentation

### 6.62.3.1 friend class Browser [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQQueueBrowser.h`



## 6.63 activemq:wireformat::openwire::marshal:v3::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v3::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.63.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.63.2 Constructor & Destructor Documentation

**6.63.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.63.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.63.3 Member Function Documentation

**6.63.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.63.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.63.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 334).

**6.63.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 334).

**6.63.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 335).

```

6.63.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 335).

```

6.63.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 336).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h`

## 6.64 activemq:wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 495).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.64.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 495).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.64.2 Constructor & Destructor Documentation

**6.64.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.64.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.64.3 Member Function Documentation

**6.64.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.64.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.64.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 338).

**6.64.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 338).

**6.64.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 339).

```

6.64.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 339).

```

6.64.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 340).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQQueueMarshaller.h**



## 6.65 activemq:wireformat::openwire::marshal:v4::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v4::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.65.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.65.2 Constructor & Destructor Documentation

**6.65.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.65.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.65.3 Member Function Documentation

**6.65.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.65.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.65.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 342).

**6.65.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 342).

**6.65.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 343).

```

6.65.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 343).

```

6.65.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 344).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

## 6.66 activemq:wireformat::openwire::marshal:v5::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p.503).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v5::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.66.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p.503).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.66.2 Constructor & Destructor Documentation

**6.66.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.66.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.66.3 Member Function Documentation

**6.66.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.66.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.66.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 346).

**6.66.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 346).

**6.66.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 347).

```

6.66.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 347).

```

6.66.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 348).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQQueueMarshaller.h**



## 6.67 activemq:wireformat::openwire::marshal:v2::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p.507).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.67.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p.507).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.67.2 Constructor & Destructor Documentation

**6.67.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.67.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.67.3 Member Function Documentation

**6.67.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.67.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.67.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 350).

**6.67.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 350).

**6.67.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 351).

```

6.67.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 351).

```

6.67.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 352).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQQueueMarshaller.h**

## 6.68 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 511).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.68.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 511).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.68.2 Constructor & Destructor Documentation

**6.68.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

**6.68.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

## 6.68.3 Member Function Documentation

**6.68.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.68.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.68.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 354).

**6.68.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 354).

**6.68.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 355).

```

6.68.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 355).

```

6.68.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 356).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h



## 6.69 activemq::core::ActiveMQSession Class Reference

#include <src/main/activemq/core/ActiveMQSession.h> Inheritance diagram for activemq::core::ActiveMQSession:

### Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** \*connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)  
*Redispatches the given set of unconsumed messages to the consumers.*
- void **start** ()  
*Stops asynchronous message delivery.*
- void **stop** ()  
*Starts asynchronous message delivery.*
- bool **isStarted** () const  
*Indicates whether or not the session is currently in the started **state** (p. 95).*
- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)  
*Fires the given exception to the exception listener of the connection.*
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)  
*Dispatches a message to a particular consumer.*
- virtual void **close** () throw ( **cms::CMSEException** )  
*Closes this session as well as any active child consumers or producers.*
- virtual void **commit** () throw ( **cms::CMSEException** )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** () throw ( **cms::CMSEException** )  
*Rollsback all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** () throw ( **cms::CMSEException** )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination) throw ( **cms::CMSEException** )

*Creates a MessageConsumer for the specified destination.*

- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector) throw ( cms::CMSEException )

*Creates a MessageConsumer for the specified destination, using a message selector.*

- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )

*Creates a MessageConsumer for the specified destination, using a message selector.*

- virtual **cms::MessageConsumer** \* **createDurableConsumer** (const **cms::Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSEException )

*Creates a durable subscriber to the specified topic, using a message selector.*

- virtual **cms::MessageProducer** \* **createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )

*Creates a MessageProducer to send messages to the specified destination.*

- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::Queue** \* **createQueue** (const std::string &queueName) throw ( cms::CMSEException )

*Creates a queue identity given a Queue name.*

- virtual **cms::Topic** \* **createTopic** (const std::string &topicName) throw ( cms::CMSEException )

*Creates a topic identity given a Queue name.*

- virtual **cms::TemporaryQueue** \* **createTemporaryQueue** () throw ( cms::CMSEException )

*Creates a TemporaryQueue object.*

- virtual **cms::TemporaryTopic** \* **createTemporaryTopic** () throw ( cms::CMSEException )

*Creates a TemporaryTopic object.*

- virtual **cms::Message** \* **createMessage** () throw ( cms::CMSEException )

*Creates a new Message.*

- virtual **cms::BytesMessage** \* **createBytesMessage** () throw ( cms::CMSEException )

*Creates a BytesMessage.*

- virtual **cms::BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSEException )  
*Creates a BytesMessage and sets the pay-load to the passed value.*
- virtual **cms::StreamMessage** \* **createStreamMessage** () throw ( cms::CMSEException )  
*Creates a new StreamMessage.*
- virtual **cms::TextMessage** \* **createTextMessage** () throw ( cms::CMSEException )  
*Creates a new TextMessage.*
- virtual **cms::TextMessage** \* **createTextMessage** (const std::string &text) throw ( cms::CMSEException )  
*Creates a new TextMessage and set the text to the value given.*
- virtual **cms::MapMessage** \* **createMapMessage** () throw ( cms::CMSEException )  
*Creates a new MapMessage.*
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw ( cms::CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const throw ( cms::CMSEException )  
*Gets if the Sessions is a Transacted Session.*
- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*
- void **send** (cms::Message \*message, ActiveMQProducer \*producer, util::Usage \*usage) throw ( cms::CMSEException )  
*Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.*
- **cms::ExceptionListener** \* **getExceptionListener** ()  
*This method gets any registered exception listener of this sessions connection and returns it.*
- const **commands::SessionInfo** & **getSessionInfo** () const  
*Gets the Session Information object for this session, if the session is closed than this method throws an exception.*
- const **commands::SessionId** & **getSessionId** () const  
*Gets the Session Id object for this session, if the session is closed than this method throws an exception.*
- **ActiveMQConnection** \* **getConnection** () const  
*Gets the **ActiveMQConnection** (p. 273) that is associated with this session.*
- long long **getLastDeliveredSequenceId** () const  
*Gets the currently set Last Delivered Sequence Id.*
- void **setLastDeliveredSequenceId** (long long value)

*Sets the value of the Last Delivered Sequence Id.*

- void **oneway** (**Pointer**< **commands::Command** > command) throw ( **activemq::exceptions::ActiveMQException** )

*Sends a oneway message.*

- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw ( **activemq::exceptions::ActiveMQException** )

*Sends a synchronous request and returns the response from the broker.*

- void **addConsumer** (**ActiveMQConsumer** \*consumer) throw ( **activemq::exceptions::ActiveMQException** )

*Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*

- void **removeConsumer** (const **Pointer**< **commands::ConsumerId** > &consumerId, long long lastDeliveredSequenceId=0) throw ( **activemq::exceptions::ActiveMQException** )

*Dispose of a MessageConsumer from this session.*

- void **addProducer** (**ActiveMQProducer** \*consumer) throw ( **activemq::exceptions::ActiveMQException** )

*Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*

- void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw ( **activemq::exceptions::ActiveMQException** )

*Dispose of a MessageProducer from this session.*

- void **doStartTransaction** () throw ( **exceptions::ActiveMQException** )

*Starts if not already start a Transaction for this Session.*

- **Pointer**< **ActiveMQTransactionContext** > **getTransactionContext** ()

*Gets the Pointer to this Session's TransactionContext.*

- void **acknowledge** ()

*Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.*

- void **deliverAcks** ()

*Request that this Session inform all of its consumers to deliver their pending acks.*

- void **clearMessagesInProgress** ()

*Request that this Session inform all of its consumers to clear all messages that are currently in progress.*

- void **wakeup** ()

*Causes the Session to wakeup its executor and ensure all messages are dispatched.*

- **Pointer**< **commands::ConsumerId** > **getNextConsumerId** ()

*Get the Next available Consumer Id.*

- `Pointer< commands::ProducerId > getNextProducerId ()`  
*Get the Next available Producer Id.*

## Friends

- class `ActiveMQSessionExecutor`

## 6.69.1 Constructor & Destructor Documentation

**6.69.1.1** `activemq::core::ActiveMQSession::ActiveMQSession (const Pointer< commands::SessionInfo > & sessionInfo, cms::Session::AcknowledgeMode ackMode, const decaf::util::Properties & properties, ActiveMQConnection * connection)`

**6.69.1.2** `virtual activemq::core::ActiveMQSession::~~ActiveMQSession ()` [virtual]

## 6.69.2 Member Function Documentation

**6.69.2.1** `void activemq::core::ActiveMQSession::acknowledge ()`

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

**6.69.2.2** `void activemq::core::ActiveMQSession::addConsumer (ActiveMQConsumer * consumer) throw ( activemq::exceptions::ActiveMQException )`

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

### Parameters:

*consumer* The `ActiveMQConsumer` (p. 310) instance to add to this session.

### Exceptions:

*ActiveMQException* if an internal error occurs.

**6.69.2.3** `void activemq::core::ActiveMQSession::addProducer (ActiveMQProducer * consumer) throw ( activemq::exceptions::ActiveMQException )`

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

### Parameters:

*consumer* The `ActiveMQProducer` (p. 470) instance to add to this session.

### Exceptions:

*ActiveMQException* if an internal error occurs.

**6.69.2.4 void activemq::core::ActiveMQSession::clearMessagesInProgress ()**

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

**6.69.2.5 virtual void activemq::core::ActiveMQSession::close () throw ( cms::CMSException ) [virtual]**

Closes this session as well as any active child consumers or producers.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3368).

**6.69.2.6 virtual void activemq::core::ActiveMQSession::commit () throw ( cms::CMSException ) [virtual]**

Commits all messages done in this transaction and releases any locks currently held.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3369).

**6.69.2.7 virtual cms::QueueBrowser\* activemq::core::ActiveMQSession::createBrowser (const cms::Queue \* *queue*, const std::string & *selector*) throw ( cms::CMSException ) [virtual]**

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters:**

*queue* the Queue to browse

*selector* the Message selector to filter which messages are browsed.

**Returns:**

New QueueBrowser that is owned by the caller.

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if the destination given is invalid.

Implements **cms::Session** (p. 3369).

**6.69.2.8** `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue) throw ( cms::CMSException )` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters:**

*queue* the Queue to browse

**Returns:**

New QueueBrowser that is owned by the caller.

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if the destination given is invalid.

Implements **cms::Session** (p. 3369).

**6.69.2.9** `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * bytes, int bytesSize) throw ( cms::CMSException )` [virtual]

Creates a BytesMessage and sets the pay-load to the passed value.

**Parameters:**

*bytes* - an array of bytes to set in the message

*bytesSize* - the size of the bytes array, or number of bytes to use

**Returns:**

a newly created BytesMessage.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3370).

**6.69.2.10** `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage () throw ( cms::CMSException )` [virtual]

Creates a BytesMessage.

**Returns:**

a newly created BytesMessage.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3370).

**6.69.2.11** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw ( cms::CMSEException ) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters:**

*destination* - The Destination that this consumer receiving messages for.

*selector* - The Message Selector string to use for this destination

*noLocal* - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Exceptions:**

*CMSEException*

Implements `cms::Session` (p. 3370).

**6.69.2.12** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw ( cms::CMSEException ) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters:**

*destination* - The Destination that this consumer receiving messages for.

*selector* - The Message Selector string to use for this destination

**Exceptions:**

*CMSEException*

Implements `cms::Session` (p. 3371).

**6.69.2.13** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination) throw ( cms::CMSEException ) [virtual]`

Creates a MessageConsumer for the specified destination.

**Parameters:**

*destination* - The Destination that this consumer receiving messages for.

**Exceptions:**

*CMSEException*

Implements `cms::Session` (p. 3371).



**6.69.2.14** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw ( cms::CMSException ) [virtual]`

Creates a durable subscriber to the specified topic, using a message selector.

**Parameters:**

*destination* - the topic to subscribe to

*name* - The name used to identify the subscription

*selector* - only messages matching the selector are received

*noLocal* - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3372).

**6.69.2.15** `virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage () throw ( cms::CMSException ) [virtual]`

Creates a new MapMessage.

**Returns:**

a newly created MapMessage.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3372).

**6.69.2.16** `virtual cms::Message* activemq::core::ActiveMQSession::createMessage () throw ( cms::CMSException ) [virtual]`

Creates a new Message.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3372).

**6.69.2.17** `virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer (const cms::Destination * destination) throw ( cms::CMSException ) [virtual]`

Creates a MessageProducer to send messages to the specified destination.

**Parameters:**

*destination* - the Destination to publish on

**Exceptions:**

*CMSException*

Implements `cms::Session` (p. 3373).

**6.69.2.18** `virtual cms::Queue* activemq::core::ActiveMQSession::createQueue (const std::string & queueName) throw ( cms::CMSException ) [virtual]`

Creates a queue identity given a Queue name.

**Parameters:**

*queueName* - the name of the new Queue

**Exceptions:**

*CMSException*

Implements `cms::Session` (p. 3373).

**6.69.2.19** `virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage () throw ( cms::CMSException ) [virtual]`

Creates a new StreamMessage.

**Returns:**

a newly created StreamMessage.

**Exceptions:**

*CMSException*

Implements `cms::Session` (p. 3373).

**6.69.2.20** `virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue () throw ( cms::CMSException ) [virtual]`

Creates a TemporaryQueue object.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3374).

**6.69.2.21** `virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic () throw ( cms::CMSException ) [virtual]`

Creates a TemporaryTopic object.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3374).

**6.69.2.22** `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & text) throw ( cms::CMSException ) [virtual]`

Creates a new TextMessage and set the text to the value given.

**Parameters:**

*text* - The initial text for the message

**Returns:**

a newly created TextMessage with the given Text set in the Message body.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3374).

**6.69.2.23** `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage () throw ( cms::CMSException ) [virtual]`

Creates a new TextMessage.

**Returns:**

a newly created TextMessage.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3374).

**6.69.2.24** `virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & topicName) throw ( cms::CMSException ) [virtual]`

Creates a topic identity given a Queue name.

**Parameters:**

*topicName* - the name of the new Topic

**Exceptions:**

*CMSException*

Implements `cms::Session` (p. 3375).

**6.69.2.25** `void activemq::core::ActiveMQSession::deliverAcks ()`

Request that this Session inform all of its consumers to deliver their pending acks.

**6.69.2.26** `virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & message) [virtual]`

Dispatches a message to a particular consumer.

**Parameters:**

*message* - the message to be dispatched

Implements `activemq::core::Dispatcher` (p. 1787).

**6.69.2.27** `void activemq::core::ActiveMQSession::doStartTransaction () throw ( exceptions::ActiveMQException )`

Starts if not already start a Transaction for this Session. If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

**Exceptions:**

*ActiveMQException* if this is not a Transacted Session.

**6.69.2.28** `void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & ex)`

Fires the given exception to the exception listener of the connection.

**6.69.2.29** `virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const throw ( cms::CMSException ) [virtual]`

Returns the acknowledgment mode of the session.

**Returns:**

the Sessions Acknowledge Mode

Implements **cms::Session** (p. 3375).

**6.69.2.30 ActiveMQConnection\* activemq::core::ActiveMQSession::getConnection  
( ) const [inline]**

Gets the **ActiveMQConnection** (p. 273) that is associated with this session.

**6.69.2.31 cms::ExceptionListener\* activemq::core::ActiveMQSession::getExceptionListener  
( )**

This method gets any registered exception listener of this sessions connection and returns it. Mainly intended for use by the objects that this session creates so that they can notify the client of **exceptions** (p. 92) that occur in the context of another thread.

**Returns:**

**cms::ExceptionListener** (p. 1838) pointer or NULL

**6.69.2.32 long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId  
( ) const [inline]**

Gets the currently set Last Delivered Sequence Id.

**Returns:**

long long containing the sequence id of the last delivered Message.

**6.69.2.33 Pointer<commands::ConsumerId> activemq::core::ActiveMQSession::getNextConsumerId  
( )**

Get the Next available Consumer Id.

**Returns:**

the next id in the sequence.

**6.69.2.34 Pointer<commands::ProducerId> activemq::core::ActiveMQSession::getNextProducerId  
( )**

Get the Next available Producer Id.

**Returns:**

the next id in the sequence.

**6.69.2.35** `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId () const`  
[inline]

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

**Returns:**

SessionId Reference

**6.69.2.36** `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo () const`  
[inline]

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

**Returns:**

SessionInfo Reference

**6.69.2.37** `Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::getTransactionContext ()`  
[inline]

Gets the Pointer to this Session's TransactionContext.

**Returns:**

a Pointer to this Session's TransactionContext

**6.69.2.38** `bool activemq::core::ActiveMQSession::isAutoAcknowledge () const`  
[inline]

References cms::Session::AUTO\_ACKNOWLEDGE.

**6.69.2.39** `bool activemq::core::ActiveMQSession::isClientAcknowledge () const`  
[inline]

References cms::Session::CLIENT\_ACKNOWLEDGE.

**6.69.2.40** `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge () const`  
[inline]

References cms::Session::DUPS\_OK\_ACKNOWLEDGE.

**6.69.2.41** `bool activemq::core::ActiveMQSession::isIndividualAcknowledge () const`  
[inline]

References cms::Session::INDIVIDUAL\_ACKNOWLEDGE.

**6.69.2.42** `bool activemq::core::ActiveMQSession::isStarted () const`

Indicates whether or not the session is currently in the started **state** (p. 95).

**6.69.2.43** `virtual bool activemq::core::ActiveMQSession::isTransacted () const  
throw ( cms::CMSException ) [virtual]`

Gets if the Sessions is a Transacted Session.

**Returns:**

transacted true - false.

Implements **cms::Session** (p. 3375).

**6.69.2.44** `void activemq::core::ActiveMQSession::oneway  
(Pointer< commands::Command > command) throw (  
activemq::exceptions::ActiveMQException )`

Sends a oneway message.

**Parameters:**

*command* The message to send.

**Exceptions:**

*ActiveMQException* if not currently connected, or if the operation fails for any reason.

**6.69.2.45** `virtual void activemq::core::ActiveMQSession::recover () throw (  
cms::CMSException ) [virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

**Exceptions:**

*CMSException* - if the CMS provider fails to stop and restart message delivery due to some internal error.

*IllegalStateException* - if the method is called by a transacted session.

Implements **cms::Session** (p. 3376).

**6.69.2.46** void activemq::core::ActiveMQSession::redispatch  
(MessageDispatchChannel & *unconsumedMessages*)

Redispatches the given set of unconsumed messages to the consumers.

**Parameters:**

*unconsumedMessages* - unconsumed messages to be redelivered.

**6.69.2.47** void activemq::core::ActiveMQSession::removeConsumer  
(const Pointer< commands::ConsumerId > & *consumerId*,  
long long *lastDeliveredSequenceId* = 0) throw (  
activemq::exceptions::ActiveMQException )

Dispose of a MessageConsumer from this session. Removes it from the Connection and clean up any resources associated with it.

**Parameters:**

*consumerId* The ConsumerId of the MessageConsumer to remove from this Session.

*lastDeliveredSequenceId* The sequenceId of the last Message the consumer delivered.

**Exceptions:**

*ActiveMQException* if an internal error occurs.

**6.69.2.48** void activemq::core::ActiveMQSession::removeProducer (const  
Pointer< commands::ProducerId > & *producerId*) throw (  
activemq::exceptions::ActiveMQException )

Dispose of a MessageProducer from this session. Removes it from the Connection and clean up any resources associated with it.

**Parameters:**

*producerId* The ProducerId of the MessageProducer to remove from this session.

**Exceptions:**

*ActiveMQException* if an internal error occurs.

**6.69.2.49** virtual void activemq::core::ActiveMQSession::rollback () throw (  
cms::CMSException ) [virtual]

Rollsback all messages done in this transaction and releases any locks currently held.

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3376).



**6.69.2.50** void activemq::core::ActiveMQSession::send (cms::Message \* *message*, ActiveMQProducer \* *producer*, util::Usage \* *usage*) throw ( cms::CMSEException )

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection. Asynchronous sends will be chosen if at all possible.

**Parameters:**

*message* The message to send to the broker.

*producer* The sending Producer

*usage* Pointer to a Usage tracker which if set will be increased by the size of the given message.

**Exceptions:**

*CMSEException*

**6.69.2.51** void activemq::core::ActiveMQSession::setLastDeliveredSequenceId (long long *value*) [inline]

Sets the value of the Last Delivered Sequence Id.

**Parameters:**

*value* The new value to assign to the Last Delivered Sequence Id property.

**6.69.2.52** void activemq::core::ActiveMQSession::start ()

Stops asynchronous message delivery.

**6.69.2.53** void activemq::core::ActiveMQSession::stop ()

Starts asynchronous message delivery.

**6.69.2.54** void activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > *command*, unsigned int *timeout* = 0) throw ( activemq::exceptions::ActiveMQException )

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

**Parameters:**

*command* The request command.

*timeout* The time to wait for a response, default is zero or infinite.

**Exceptions:**

*ActiveMQException* thrown if an error response was received from the broker, or if any other error occurred.

**6.69.2.55**    `virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & name) throw ( cms::CMSException )` [virtual]

Unsubscribes a durable subscription that has been created by a client. This method deletes the **state** (p. 95) being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

**Parameters:**

*name* the name used to identify this subscription

**Exceptions:**

*CMSException*

Implements **cms::Session** (p. 3376).

**6.69.2.56**    `void activemq::core::ActiveMQSession::wakeup ()`

Causes the Session to wakeup its executor and ensure all messages are dispatched.

## 6.69.3 Friends And Related Function Documentation

**6.69.3.1**    `friend class ActiveMQSessionExecutor` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSession.h`

## 6.70 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

#include <src/main/activemq/core/ActiveMQSessionExecutor.h> Inheritance diagram for activemq::core::ActiveMQSessionExecutor:

### Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** \*session)  
*Creates an un-started executor for the given session.*
- virtual **~ActiveMQSessionExecutor** ()  
*Calls **stop()** (p. 536) then **clear()** (p. 534).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **clearMessagesInProgress** ()  
*Removes all messages in the Dispatch Channel so that non are delivered.*
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()  
*wakeup this executor and dispatch any pending messages.*
- virtual void **start** ()  
*Starts the dispatching.*
- virtual void **stop** ()  
*Stops dispatching.*
- virtual void **close** ()  
*Terminates the dispatching thread.*
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()  
*Removes all queued messages and destroys them.*
- virtual bool **iterate** ()  
*Iterates on the **MessageDispatchChannel** (p. 2599) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

### 6.70.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

### 6.70.2 Constructor & Destructor Documentation

#### 6.70.2.1 `activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor (ActiveMQSession * session)`

Creates an un-started executor for the given session.

#### 6.70.2.2 `virtual activemq::core::ActiveMQSessionExecutor::~~ActiveMQSessionExecutor () [virtual]`

Calls `stop()` (p. 536) then `clear()` (p. 534).

### 6.70.3 Member Function Documentation

#### 6.70.3.1 `virtual void activemq::core::ActiveMQSessionExecutor::clear () [inline, virtual]`

Removes all queued messages and destroys them.

#### 6.70.3.2 `virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress () [inline, virtual]`

Removes all messages in the Dispatch Channel so that non are delivered.

#### 6.70.3.3 `virtual void activemq::core::ActiveMQSessionExecutor::close () [inline, virtual]`

Terminates the dispatching thread. Once this is called, the executor is no longer usable.

#### 6.70.3.4 `virtual void activemq::core::ActiveMQSessionExecutor::execute (const Pointer< MessageDispatch > & data) [virtual]`

Executes the dispatch. Adds the given data to the end of the queue.

**Parameters:**

*data* - the data to be dispatched.

#### 6.70.3.5 `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst (const Pointer< MessageDispatch > & data) [virtual]`

Executes the dispatch. Adds the given data to the beginning of the queue.

**Parameters:**

*data* - the data to be dispatched.

**6.70.3.6** `std::vector< Pointer<MessageDispatch> > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ()`  
[inline]

**Returns:**

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

**6.70.3.7** `virtual bool activemq::core::ActiveMQSessionExecutor::hasUnconsumedMessages ()`  
`const` [inline, virtual]

**Returns:**

true if there are any pending messages in the dispatch channel.

**6.70.3.8** `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ()`  
[inline, virtual]

**Returns:**

true if there are no messages in the Dispatch Channel.

**6.70.3.9** `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ()` `const`  
[inline, virtual]

**Returns:**

true indicates if the executor is started

**6.70.3.10** `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ()`  
[virtual]

Iterates on the **MessageDispatchChannel** (p. 2599) sending all pending messages to the Consumers they are destined for.

**Returns:**

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 3734).

**6.70.3.11** `virtual void activemq::core::ActiveMQSessionExecutor::start ()`  
[virtual]

Starts the dispatching.

**6.70.3.12** `virtual void activemq::core::ActiveMQSessionExecutor::stop ()` [virtual]

Stops dispatching.

**6.70.3.13** `virtual void activemq::core::ActiveMQSessionExecutor::wakeup ()`  
[virtual]

wakeup this executer and dispatch any pending messages.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSessionExecutor.h`

## 6.71 activemq::commands::ActiveMQStreamMessage Class Reference

#include <src/main/activemq/commands/ActiveMQStreamMessage.h> Inheritance diagram for activemq::commands::ActiveMQStreamMessage:

### Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQStreamMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual **cms::StreamMessage \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **reset** () throw ( cms::CMSException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*

- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the Stream message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the Stream message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the Stream message stream.*
- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Stream message stream.*
- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a char to the Stream message stream as a 1-byte value.*
- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Stream message stream.*
- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the Stream message stream as a 4 byte value.*
- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Stream message stream.*



- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a double to the Stream message stream as a 8 byte value.*
- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a 16 bit signed short from the Stream message stream.*
- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a signed short to the Stream message stream as a 2 byte value.*
- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a 16 bit unsigned short from the Stream message stream.*
- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a unsigned short to the Stream message stream as a 2 byte value.*
- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a 32 bit signed integer from the Stream message stream.*
- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a signed int to the Stream message stream as a 4 byte value.*
- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a 64 bit long from the Stream message stream.*
- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a long long to the Stream message stream as a 8 byte value.*
- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads an ASCII String from the Stream message stream.*
- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes an ASCII String to the Stream message stream.*

## Static Public Attributes

- static const unsigned char `ID_ACTIVEMQSTREAMMESSAGE` = 27

### 6.71.1 Constructor & Destructor Documentation

**6.71.1.1** `activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage()`

**6.71.1.2** `virtual activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage()` [virtual]

### 6.71.2 Member Function Documentation

**6.71.2.1** `virtual void activemq::commands::ActiveMQStreamMessage::clearBody () throw ( cms::CMSException )` [virtual]

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 428).

**6.71.2.2** `virtual cms::StreamMessage* activemq::commands::ActiveMQStreamMessage::clone () const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns:

new copy of this message

Implements `cms::Message` (p. 2539).

**6.71.2.3** `virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2520).

**6.71.2.4** `virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2521).

#### 6.71.2.5 virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure \* value) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 429).

#### 6.71.2.6 virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Message** (p. 2523).

#### 6.71.2.7 virtual void activemq::commands::ActiveMQStreamMessage::onSend () [virtual]

Store the Data that was written to the stream before a send.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 436).

#### 6.71.2.8 virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]

Reads a Boolean from the Stream message stream.

**Returns:**

boolean value from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3655).

**6.71.2.9** `virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte ()  
const throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a Byte from the Stream message stream.

**Returns:**

unsigned char value from stream

**Exceptions:**

*CMSEException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3655).

**6.71.2.10** `virtual int activemq::commands::ActiveMQStreamMessage::readBytes  
(unsigned char * buffer, int length) const throw (  
cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a portion of the Stream message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an CMSEException is thrown. No bytes will be read from the stream for this exception case.

**Parameters:**

*buffer* the buffer into which the data is read

*length* the number of bytes to read; must be less than or equal to value.length

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3656).

**6.71.2.11** `virtual int activemq::commands::ActiveMQStreamMessage::readBytes  
(std::vector< unsigned char > & value) const throw (  
cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a byte array from the Stream message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

**Parameters:**

*value* buffer to place data in

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3656).

**6.71.2.12** `virtual char activemq::commands::ActiveMQStreamMessage::readChar  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads a Char from the Stream message stream.

**Returns:**

char value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3657).

**6.71.2.13** `virtual double activemq::commands::ActiveMQStreamMessage::readDouble() const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 64 bit double from the Stream message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3657).

**6.71.2.14** `virtual float activemq::commands::ActiveMQStreamMessage::readFloat() const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 32 bit float from the Stream message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3658).

**6.71.2.15** `virtual int activemq::commands::ActiveMQStreamMessage::readInt  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads a 32 bit signed integer from the Stream message stream.

**Returns:**

int value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3658).

**6.71.2.16** `virtual long long ac-  
tivismq::commands::ActiveMQStreamMessage::readLong  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads a 64 bit long from the Stream message stream.

**Returns:**

long long value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3658).

**6.71.2.17** `virtual short activemq::commands::ActiveMQStreamMessage::readShort  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads a 16 bit signed short from the Stream message stream.

**Returns:**

short value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3659).

```
6.71.2.18 virtual std::string ac-
tivismq::commands::ActiveMQStreamMessage::readString
() const throw ( cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException ) [virtual]
```

Reads an ASCII String from the Stream message stream.

**Returns:**

String from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3659).

```
6.71.2.19 virtual unsigned short ac-
tivismq::commands::ActiveMQStreamMessage::readUnsignedShort
() const throw ( cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException ) [virtual]
```

Reads a 16 bit unsigned short from the Stream message stream.

**Returns:**

unsigned short value from stream

**Exceptions:**

*CMSException* - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* - if unexpected end of message stream has been reached.

*MessageFormatException* - if this type conversion is invalid.

*MessageNotReadableException* - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3660).



**6.71.2.20** virtual void activemq::commands::ActiveMQStreamMessage::reset ()  
throw ( cms::CMSException ) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

**Exceptions:**

*CMSException*

**6.71.2.21** virtual std::string activemq::commands::ActiveMQStreamMessage::toString ()  
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2530).

**6.71.2.22** virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool *value*)  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]

Writes a boolean to the Stream message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters:**

*value* boolean to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3660).

**6.71.2.23** virtual void activemq::commands::ActiveMQStreamMessage::writeByte  
(unsigned char *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a byte to the Stream message stream as a 1-byte value.

**Parameters:**

*value* byte to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3660).

**6.71.2.24** `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Writes a portion of a byte array to the Stream message stream. size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

*offset* the initial offset within the byte array

*length* the number of bytes to use

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3661).

**6.71.2.25** `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const std::vector< unsigned char > & value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3661).

**6.71.2.26** `virtual void activemq::commands::ActiveMQStreamMessage::writeChar (char value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Writes a char to the Stream message stream as a 1-byte value.

**Parameters:**

*value* char to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3661).

**6.71.2.27** virtual void activemq::commands::ActiveMQStreamMessage::writeDouble  
(double *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a double to the Stream message stream as a 8 byte value.

**Parameters:**

*value* double to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3662).

**6.71.2.28** virtual void activemq::commands::ActiveMQStreamMessage::writeFloat  
(float *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a float to the Stream message stream as a 4 byte value.

**Parameters:**

*value* float to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3662).

**6.71.2.29** virtual void activemq::commands::ActiveMQStreamMessage::writeInt  
(int *value*) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

**Parameters:**

*value* signed int to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3662).

**6.71.2.30**    `virtual void activemq::commands::ActiveMQStreamMessage::writeLong  
(long long value) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

**Parameters:**

*value* signed long long to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3663).

**6.71.2.31**    `virtual void activemq::commands::ActiveMQStreamMessage::writeShort  
(short value) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

**Parameters:**

*value* signed short to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3663).

**6.71.2.32**    `virtual void activemq::commands::ActiveMQStreamMessage::writeString  
(const std::string & value) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]`

Writes an ASCII String to the Stream message stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3663).

**6.71.2.33** `virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

**Parameters:**

*value* unsigned short to write to the stream

**Exceptions:**

*CMSException* - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3664).

### 6.71.3 Field Documentation

**6.71.3.1** `const unsigned char activemq::commands::ActiveMQStreamMessage::ID_ - ACTIVEMQSTREAMMESSAGE = 27 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQStreamMessage.h`

## 6.72 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 552).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.72.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 552). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.72.2 Constructor & Destructor Documentation

**6.72.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` `[inline]`

**6.72.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller()` `[inline, virtual]`

## 6.72.3 Member Function Documentation

**6.72.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.72.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.72.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.72.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699).

**6.72.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700).



## 6.72

activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller

Class Reference

555

```
6.72.3.6 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMars
      (OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataOutputStream * dataOut,
      utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
(p. 2701).

```
6.72.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnm
      (OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
      * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
(p. 2701).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**

## 6.73 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 556).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.73.1 Detailed Description

Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 556). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.73.2 Constructor & Destructor Documentation

**6.73.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` `[inline]`

**6.73.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller()` `[inline, virtual]`

## 6.73.3 Member Function Documentation

**6.73.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject() const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.73.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.73.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.73.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714).

**6.73.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2715).

```

6.73.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMars
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

```

6.73.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnm
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2716).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQStreamMessageMarshaller.h**

## 6.74 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 560).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.74.1 Detailed Description

Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 560). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.74.2 Constructor & Destructor Documentation

**6.74.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

**6.74.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller()` [inline, virtual]

## 6.74.3 Member Function Documentation

**6.74.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.74.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.74.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.74.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709).

**6.74.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2710).



```

6.74.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMars
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

```

6.74.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnm
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2711).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQStreamMessageMarshaller.h**

## 6.75 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 564).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.75.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 564). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.75.2 Constructor & Destructor Documentation

**6.75.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` `[inline]`

**6.75.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` `[inline, virtual]`

## 6.75.3 Member Function Documentation

**6.75.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.75.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.75.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.75.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694).

**6.75.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2695).

```

6.75.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMars
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

```

6.75.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnm
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQStreamMessageMarshaller.h**

## 6.76 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 568).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.76.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 568). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.76.2 Constructor & Destructor Documentation

**6.76.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` `[inline]`

**6.76.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller()` `[inline, virtual]`

## 6.76.3 Member Function Documentation

**6.76.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject() const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.76.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.76.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.76.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2704).

**6.76.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705).



## 6.76

activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller

Class Reference

571

```
6.76.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMars
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
(p. 2706).

```
6.76.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnm
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
(p. 2706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**

## 6.77 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 572).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.77.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 572). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.77.2 Constructor & Destructor Documentation

**6.77.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

**6.77.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

## 6.77.3 Member Function Documentation

**6.77.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.77.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.77.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.77.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719).

**6.77.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720).

## 6.77

activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller

Class Reference

575

```
6.77.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMars
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

```
6.77.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightUnm
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQStreamMessageMarshaller.h**

## 6.78 activemq::commands::ActiveMQTempDestination Class Reference

#include <src/main/activemq/commands/ActiveMQTempDestination.h> Inheritance diagram for activemq::commands::ActiveMQTempDestination:

### Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempDestination** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **close** () throw ( cms::CMSException )  
*Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.*
- void **setConnection** (core::ActiveMQConnection \*connection)  
*Sets the Parent Connection that is notified when this destination is destroyed.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPDESTINATION** = 0

### Protected Attributes

- core::ActiveMQConnection \* **connection**  
*Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.*

## 6.78.1 Constructor & Destructor Documentation

- 6.78.1.1** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination()`
- 6.78.1.2** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination(const std::string & name)`
- 6.78.1.3** `virtual  
activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination()  
()` [virtual]

## 6.78.2 Member Function Documentation

- 6.78.2.1** `virtual ActiveMQTempDestination* ac-  
tivemq::commands::ActiveMQTempDestination::cloneDataStructure ()  
const` [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 324).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 605), and `activemq::commands::ActiveMQTempTopic` (p. 634).

- 6.78.2.2** `virtual void activemq::commands::ActiveMQTempDestination::close ()  
throw ( cms::CMSException )` [virtual]

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker. This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws `cms::CMSException` (p. 1160)

Implements `cms::Closeable` (p. 1148).

- 6.78.2.3** `virtual void ac-  
tivemq::commands::ActiveMQTempDestination::copyDataStructure (const  
DataStructure * src)` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns:

src - Source Object

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 325).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 606), and `activemq::commands::ActiveMQTempTopic` (p. 635).

References `activemq::commands::ActiveMQDestination::copyDataStructure()`.

#### 6.78.2.4 `virtual bool activemq::commands::ActiveMQTempDestination::equals` `(const DataStructure * value) const` [inline, virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

##### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 326).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 606), and `activemq::commands::ActiveMQTempTopic` (p. 635).

References `activemq::commands::ActiveMQDestination::equals()`.

#### 6.78.2.5 `virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType` `() const` [virtual]

Get the **DataStructure** (p. 1660) Type as defined in `CommandTypes.h`.

##### Returns:

The type of the data structure

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 326).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 607), and `activemq::commands::ActiveMQTempTopic` (p. 636).

#### 6.78.2.6 `void activemq::commands::ActiveMQTempDestination::setConnection` `(core::ActiveMQConnection * connection)` [inline]

Sets the Parent Connection that is notified when this destination is destroyed.

##### Parameters:

*connection* - The parent connection.

#### 6.78.2.7 `virtual std::string activemq::commands::ActiveMQTempDestination::toString` `() const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

##### Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 330).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 607), and `activemq::commands::ActiveMQTempTopic` (p. 636).



### 6.78.3 Field Documentation

#### 6.78.3.1 `core::ActiveMQConnection*` `activemq::commands::ActiveMQTempDestination::connection` [protected]

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

#### 6.78.3.2 `const unsigned char` `activemq::commands::ActiveMQTempDestination::ID_` - `ACTIVEMQTEMPDESTINATION = 0` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

## 6.79 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 580).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>  
 diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.79.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 580). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.79.2 Constructor & Destructor Documentation

**6.79.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.79.2.2** `virtual ~ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.79.3 Member Function Documentation

**6.79.3.1** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 334).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 610), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 639).

**6.79.3.2** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 334).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 611), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 640).

**6.79.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 335).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 611), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 640).

**6.79.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.79

**activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

**Class Reference**

**583**

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 335).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 611), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 640).

### 6.79.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 336).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 612), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 641).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h`

## 6.80 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 584).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>  
 diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.80.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 584). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.80.2 Constructor & Destructor Documentation

**6.80.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.80.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.80.3 Member Function Documentation

**6.80.3.1** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 338).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 614), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 647).

**6.80.3.2** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 338).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 615), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 648).

**6.80.3.3** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshall(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 339).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 615), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 648).

**6.80.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshall(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.80

**activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

**Class Reference**

587

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 339).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 615), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 648).

### 6.80.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 340).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 616), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 649).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h`

## 6.81 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 588).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>  
 diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.81.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 588). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.81.2 Constructor & Destructor Documentation

**6.81.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.81.2.2** `virtual ~ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.81.3 Member Function Documentation

**6.81.3.1** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 342).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 618), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 643).

**6.81.3.2** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 342).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 619), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 644).

**6.81.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshall(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 343).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 619), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 644).

**6.81.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshall(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.81

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

**Class Reference**

591

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 343).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 619), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 644).

### 6.81.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 344).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 620), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 645).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**

## 6.82 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 592).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>  
 diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.82.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 592). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.82.2 Constructor & Destructor Documentation

**6.82.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.82.2.2** `virtual ~ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.82.3 Member Function Documentation

**6.82.3.1** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 346).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 622), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 651).

**6.82.3.2** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 346).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 623), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 652).

**6.82.3.3** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 347).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 623), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 652).

**6.82.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.82

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

**Class Reference**

595

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 347).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 623), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 652).

### 6.82.3.5 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 348).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 624), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 653).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempDestinationMarshaller.h**

## 6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 596).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>  
 diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.83.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 596). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.83.2 Constructor & Destructor Documentation

**6.83.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.83.2.2** `virtual ~ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.83.3 Member Function Documentation

**6.83.3.1** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 350).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 626), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 655).

**6.83.3.2** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the **unmarshal**.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 350).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 627), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 656).

**6.83.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 351).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 627), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 656).

**6.83.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.83

**activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

**Class Reference**

599

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 351).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 627), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 656).

### 6.83.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 352).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 628), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 657).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h`

## 6.84 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDesti Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 600).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>  
diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.84.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 600). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.84.2 Constructor & Destructor Documentation

**6.84.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` `[inline]`

**6.84.2.2** `virtual ~activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller()` `[inline, virtual]`

## 6.84.3 Member Function Documentation

**6.84.3.1** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 354).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 630), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 659).

**6.84.3.2** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` `[virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 354).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 631), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 660).

**6.84.3.3** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 355).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 631), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 660).

**6.84.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).



## 6.84

**activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

**Class Reference**

**603**

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 355).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 631), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 660).

### 6.84.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightUn  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream  
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 356).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 632), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 661).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h`

## 6.85 activemq::commands::ActiveMQTempQueue Class Reference

#include <src/main/activemq/commands/ActiveMQTempQueue.h> Inheritance diagram for activemq::commands::ActiveMQTempQueue:

### Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempQueue** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Converts the Destination Name into a String.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destinastion object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getQueueName** () const throw ( cms::CMSException )  
*Gets the name of this queue.*
- virtual void **destroy** () throw ( cms::CMSException )  
*Destroy's the Temp Destination at the Broker.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPQUEUE** = 102

## 6.85.1 Constructor & Destructor Documentation

**6.85.1.1** `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ()`

**6.85.1.2** `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue (const std::string & name)`

**6.85.1.3** `virtual  
activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue ()  
[virtual]`

## 6.85.2 Member Function Documentation

**6.85.2.1** `virtual cms::Destination* ac-  
tivemq::commands::ActiveMQTempQueue::clone () const  
[inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1724).

**6.85.2.2** `virtual ActiveMQTempQueue* ac-  
tivemq::commands::ActiveMQTempQueue::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 577).

**6.85.2.3** `virtual void activemq::commands::ActiveMQTempQueue::copy (const  
cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

### Parameters:

*source* The source Destination object.

**6.85.2.4** `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 577).

**6.85.2.5** `virtual void activemq::commands::ActiveMQTempQueue::destroy () throw ( cms::CMSException ) [virtual]`

Destroy's the Temp Destination at the Broker.

**Exceptions:**

*CMSException*

Implements `cms::TemporaryQueue` (p. 3759).

**6.85.2.6** `virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 578).

**6.85.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination () const [inline, virtual]`

**Returns:**

the `cms::Destination` (p. 1723) interface pointer that the objects that derive from this class implement.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 326).

**6.85.2.8** `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns:**

const reference to a properties object.

Implements **cms::Destination** (p. 1724).

**6.85.2.9** `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

**Returns:**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 578).

**6.85.2.10** `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

**Returns:**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 327).

References cms::Destination::TEMPORARY\_QUEUE.

**6.85.2.11** `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this queue.

**Returns:**

The queue name.

Implements **cms::TemporaryQueue** (p. 3760).

**6.85.2.12** `virtual std::string activemq::commands::ActiveMQTempQueue::toString () const [virtual]`

Converts the Destination Name into a String.

**Returns:**

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 578).

### 6.85.3 Field Documentation

#### 6.85.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_ - ACTIVEMQTEMPQUEUE = 102` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

## 6.86 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 609).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.86.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 609). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.86.2 Constructor & Destructor Documentation

**6.86.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.86.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.86.3 Member Function Documentation

**6.86.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject(const std::string&)` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.86.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType()` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.86.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581).

**6.86.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581).

**6.86.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 582).

**6.86.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 582).

**6.86.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 583).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h`

## 6.87 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.613).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.87.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.613). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.87.2 Constructor & Destructor Documentation

**6.87.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.87.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.87.3 Member Function Documentation

**6.87.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.87.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.87.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 585).

**6.87.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 585).

**6.87.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 586).

**6.87.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 586).

**6.87.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 587).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h`

## 6.88 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.617).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.88.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.617). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.88.2 Constructor & Destructor Documentation

**6.88.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.88.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.88.3 Member Function Documentation

**6.88.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.88.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.88.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 589).

**6.88.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 589).

**6.88.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 590).

**6.88.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 590).

**6.88.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 591).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h`

## 6.89 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.621).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.89.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.621). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.89.2 Constructor & Destructor Documentation

**6.89.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.89.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.89.3 Member Function Documentation

**6.89.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.89.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.89.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 593).

**6.89.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 593).

**6.89.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 594).

**6.89.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 594).

**6.89.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 595).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h`

## 6.90 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.625).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.90.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p.625). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.90.2 Constructor & Destructor Documentation

**6.90.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.90.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.90.3 Member Function Documentation

**6.90.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.90.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.90.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 597).

**6.90.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 597).

**6.90.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 598).

**6.90.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 598).

**6.90.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 599).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h`

## 6.91 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 629).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.91.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 629). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.91.2 Constructor & Destructor Documentation

**6.91.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

**6.91.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

## 6.91.3 Member Function Documentation

**6.91.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.91.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.91.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 601).

**6.91.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 601).

**6.91.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 602).

```

6.91.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 602).

```

6.91.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightUnmarsh
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 603).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempQueueMarshaller.h**

## 6.92 activemq::commands::ActiveMQTempTopic Class Reference

#include <src/main/activemq/commands/ActiveMQTempTopic.h> Inheritance diagram for activemq::commands::ActiveMQTempTopic:

### Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual **~ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempTopic \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Converts the Destination Name into a String.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destinastion object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getTopicName** () const throw ( cms::CMSException )  
*Gets the name of this topic.*
- virtual void **destroy** () throw ( cms::CMSException )  
*Destroy's the Temp Destination at the Broker.*

## Static Public Attributes

- static const unsigned char **ID\_**ACTIVEMQTEMPTOPIC = 103

## 6.92.1 Constructor & Destructor Documentation

**6.92.1.1** `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()`

**6.92.1.2** `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & name)`

**6.92.1.3** `virtual  
activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic ()  
[virtual]`

## 6.92.2 Member Function Documentation

**6.92.2.1** `virtual cms::Destination* ac-  
tivemq::commands::ActiveMQTempTopic::clone () const  
[inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1724).

**6.92.2.2** `virtual ActiveMQTempTopic* ac-  
tivemq::commands::ActiveMQTempTopic::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 577).

**6.92.2.3** `virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

### Parameters:

***source*** The source Destination object.



**6.92.2.4** `virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 577).

**6.92.2.5** `virtual void activemq::commands::ActiveMQTempTopic::destroy () throw ( cms::CMSException ) [virtual]`

Destroy's the Temp Destination at the Broker.

**Exceptions:**

*CMSException*

Implements `cms::TemporaryTopic` (p. 3761).

**6.92.2.6** `virtual bool activemq::commands::ActiveMQTempTopic::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 578).

**6.92.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination () const [inline, virtual]`

**Returns:**

the **cms::Destination** (p. 1723) interface pointer that the objects that derive from this class implement.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 326).

**6.92.2.8** `virtual const cms::CMSPProperties& activemq::commands::ActiveMQTempTopic::getCMSPProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns:**

const reference to a properties object.

Implements **cms::Destination** (p. 1724).

**6.92.2.9** `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

**Returns:**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 578).

**6.92.2.10** `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

**Returns:**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 327).

References cms::Destination::TEMPORARY\_TOPIC.

**6.92.2.11** `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this topic.

**Returns:**

The topic name.

Implements **cms::TemporaryTopic** (p. 3762).

**6.92.2.12** `virtual std::string activemq::commands::ActiveMQTempTopic::toString () const [virtual]`

Converts the Destination Name into a String.

**Returns:**

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 578).

### 6.92.3 Field Documentation

#### 6.92.3.1 `const unsigned char activemq::commands::ActiveMQTempTopic::ID_- ACTIVEMQTEMPTOPIC = 103` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempTopic.h`

## 6.93 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopic Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.638).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.93.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.638). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.93.2 Constructor & Destructor Documentation

**6.93.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.93.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.93.3 Member Function Documentation

**6.93.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.93.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.93.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581).

**6.93.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 581).

**6.93.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 582).

**6.93.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 582).

**6.93.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 583).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h`

## 6.94 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopic Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.642).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.94.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.642). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.94.2 Constructor & Destructor Documentation

**6.94.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.94.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.94.3 Member Function Documentation

**6.94.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.94.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.94.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 589).

**6.94.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 589).

**6.94.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 590).

**6.94.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 590).

**6.94.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 591).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h`

## 6.95 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopic Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.646).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.95.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.646). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.95.2 Constructor & Destructor Documentation

**6.95.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.95.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.95.3 Member Function Documentation

**6.95.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.95.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.95.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 585).

**6.95.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 585).

**6.95.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 586).

**6.95.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 586).

**6.95.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 587).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h`

## 6.96 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopic Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.650).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.96.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p.650). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.96.2 Constructor & Destructor Documentation

**6.96.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.96.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.96.3 Member Function Documentation

**6.96.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.96.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.96.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 593).

**6.96.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 593).

**6.96.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 594).

**6.96.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 594).

**6.96.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 595).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h`

## 6.97 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopic Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.97.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.97.2 Constructor & Destructor Documentation

**6.97.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.97.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.97.3 Member Function Documentation

**6.97.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.97.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.97.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 597).

**6.97.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 597).

**6.97.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 598).

**6.97.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 598).

**6.97.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 599).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h`

## 6.98 activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopic Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.98.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.98.2 Constructor & Destructor Documentation

**6.98.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` `[inline]`

**6.98.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` `[inline, virtual]`

## 6.98.3 Member Function Documentation

**6.98.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.98.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.98.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 601).

**6.98.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 601).

**6.98.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 602).

**6.98.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 602).

**6.98.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 603).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h`

## 6.99 activemq::commands::ActiveMQTextMessage Class Reference

#include <src/main/activemq/commands/ActiveMQTextMessage.h> Inheritance diagram for activemq::commands::ActiveMQTextMessage:

### Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQTextMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **beforeMarshal** (wireformat::WireFormat \*wireFormat) throw ( decaf::io::IOException )  
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2516) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const  
*Returns the Size of this message in Bytes.*
- virtual cms::TextMessage \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::string **getText** () const throw ( cms::CMSException )  
*Gets the message character buffer.*
- virtual void **setText** (const char \*msg) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*

- virtual void **setText** (const std::string &msg) throw ( cms::MessageNotWriteableException, cms::CMSException )

*Sets the message contents.*

## Data Fields

- std::auto\_ptr< std::string > **text**

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEXTMESSAGE** = 28

### 6.99.1 Constructor & Destructor Documentation

**6.99.1.1** `activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ()`

**6.99.1.2** `virtual  
activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage ()  
[virtual]`

### 6.99.2 Member Function Documentation

**6.99.2.1** `virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal  
(wireformat::WireFormat * wireFormat) throw ( decaf::io::IOException )  
[virtual]`

Performs any cleanup or other tasks that must be done before the **Message** (p. 2516) is marshalled to its binary WireFormat version.

#### Parameters:

*wireFormat* the WireFormat instance that is marshalling this message.

Implements `activemq::wireformat::MarshalAware` (p. 2485).

**6.99.2.2** `virtual void activemq::commands::ActiveMQTextMessage::clearBody ()  
throw ( cms::CMSException ) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 428).

**6.99.2.3** `virtual cms::TextMessage* ac-  
tivemq::commands::ActiveMQTextMessage::clone () const  
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

**Returns:**

new copy of this message

Implements **cms::Message** (p. 2539).

**6.99.2.4** `virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2520).

**6.99.2.5** `virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2521).

**6.99.2.6** `virtual bool activemq::commands::ActiveMQTextMessage::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 429).

**6.99.2.7** `virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Message** (p. 2523).

**6.99.2.8 virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize  
( ) const [virtual]**

Returns the Size of this message in Bytes.

**Returns:**

number of bytes this message equates to.

Reimplemented from **activemq::commands::Message** (p. 2525).

**6.99.2.9 virtual std::string activemq::commands::ActiveMQTextMessage::getText  
( ) const throw ( cms::CMSException ) [virtual]**

Gets the message character buffer.

**Returns:**

The message character buffer.

**Exceptions:**

*CMSException* - if an internal error occurs.

Implements **cms::TextMessage** (p. 3763).

**6.99.2.10 virtual void activemq::commands::ActiveMQTextMessage::setText  
(const std::string & msg) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]**

Sets the message contents.

**Parameters:**

*msg* The message buffer.

**Exceptions:**

*CMSException* - if an internal error occurs.

*MessageNotWriteableException* - if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3764).

**6.99.2.11 virtual void activemq::commands::ActiveMQTextMessage::setText  
(const char \* msg) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]**

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

**Parameters:**

*msg* The message buffer.

**Exceptions:**

*CMSException* - if an internal error occurs.

*MessageNotWriteableException* - if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3764).

**6.99.2.12** `virtual std::string activemq::commands::ActiveMQTextMessage::toString()  
() const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2530).

**6.99.3 Field Documentation**

**6.99.3.1** `const unsigned char activemq::commands::ActiveMQTextMessage::ID_-  
ACTIVEMQTEXTMESSAGE = 28 [static]`

**6.99.3.2** `std::auto_ptr<std::string> ac-  
tivemq::commands::ActiveMQTextMessage::text  
[mutable]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTextMessage.h`



## 6.100

**activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

Class Reference

**6.100** **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** <sup>667</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.667).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h> Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.100.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.667). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.100.2 Constructor & Destructor Documentation

**6.100.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.100.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.100.3 Member Function Documentation

**6.100.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.100.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.100.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.100

**activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

### Class Reference

669

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2699).

**6.100.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2699).

**6.100.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2700).

**6.100.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

**6.100.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2701).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h`

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.671).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.101.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.671). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.101.2 Constructor & Destructor Documentation

**6.101.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.101.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.101.3 Member Function Documentation

**6.101.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.101.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.101.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.101

**activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

**Class Reference**

**673**

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2709).

**6.101.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmar**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

### Parameters:

***wireFormat*** - describes the wire format of the broker

***dataStructure*** - Object to be marshaled

***dataIn*** - BinaryReader that provides that data source

### Exceptions:

***IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2709).

**6.101.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarsh**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

### Parameters:

***wireFormat*** - describes the wire format of the broker

***dataStructure*** - Object to be marshaled

***bs*** - BooleanStream stream used to pack bits from the wire.

### Returns:

int value indicating the size of the marshaled object.

### Exceptions:

***IOException*** if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2710).

**6.101.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p.2711).

**6.101.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p.2711).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h



## 6.102

**activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

Class Reference

**6.102** **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** <sup>675</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.675).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h> Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.102.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.675). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.102.2 Constructor & Destructor Documentation

**6.102.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.102.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.102.3 Member Function Documentation

**6.102.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.102.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.102.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.102

**activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

**Class Reference**

**677**

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2714).

**6.102.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmar**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2714).

**6.102.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarsh**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \***  
***dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

### Returns:

int value indicating the size of the marshaled object.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2715).

**6.102.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p.2716).

**6.102.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p.2716).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTextMessageMarshaller.h**

## 6.103

**activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

Class Reference

**6.103** **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** <sup>679</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.679).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h> Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.103.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.679). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.103.2 Constructor & Destructor Documentation

**6.103.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.103.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.103.3 Member Function Documentation

**6.103.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.103.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.103.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.103

**activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

### Class Reference

681

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2694).

**6.103.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2694).

**6.103.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2695).

**6.103.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

**6.103.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2696).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTextMessageMarshaller.h**



## 6.104

**activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

Class Reference

**6.104** **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** <sup>683</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.683).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h> Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.104.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.683). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the **activemq-openwire-generator** module

## 6.104.2 Constructor & Destructor Documentation

**6.104.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.104.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.104.3 Member Function Documentation

**6.104.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.104.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.104.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.104

**activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

**Class Reference**

685

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2719).

**6.104.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseUnmar**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2719).

**6.104.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarsh**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \***  
***dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

### Returns:

int value indicating the size of the marshaled object.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2720).

**6.104.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

**6.104.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2721).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h`

6.105

activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller

Class Reference

6.105 ~~activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller~~<sup>687</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.687).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.105.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.687). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.105.2 Constructor & Destructor Documentation

**6.105.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

**6.105.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

## 6.105.3 Member Function Documentation

**6.105.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.105.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.105.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.105

**activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

### Class Reference

689

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2704).

**6.105.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2704).

**6.105.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2705).

**6.105.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

**6.105.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h`



## 6.106 activemq::commands::ActiveMQTopic Class Reference

#include <src/main/activemq/commands/ActiveMQTopic.h> Inheritance diagram for activemq::commands::ActiveMQTopic:

### Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getTopicName** () const throw ( cms::CMSException )  
*Gets the name of this topic.*

### Static Public Attributes

- static const unsigned char **ID \_ACTIVEMQTOPIC** = 101

## 6.106.1 Constructor & Destructor Documentation

**6.106.1.1** `activemq::commands::ActiveMQTopic::ActiveMQTopic ()`

**6.106.1.2** `activemq::commands::ActiveMQTopic::ActiveMQTopic (const std::string & name)`

**6.106.1.3** `virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic ()`  
[virtual]

## 6.106.2 Member Function Documentation

**6.106.2.1** `virtual cms::Destination* activemq::commands::ActiveMQTopic::clone ()`  
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns:

cloned copy of this object

Implements **cms::Destination** (p.1724).

**6.106.2.2** `virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p.324).

**6.106.2.3** `virtual void activemq::commands::ActiveMQTopic::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

### Parameters:

*source* The source Destination object.

**6.106.2.4** `virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p.325).

**6.106.2.5 virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure \* *value*) const** [inline, virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

References **activemq::commands::ActiveMQDestination::equals()**.

**6.106.2.6 virtual const cms::Destination\* activemq::commands::ActiveMQTopic::getCMSDestination () const** [inline, virtual]**Returns:**

the **cms::Destination** (p. 1723) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

**6.106.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const** [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns:**

const reference to a properties object.

Implements **cms::Destination** (p. 1724).

**6.106.2.8 virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const** [virtual]

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

**Returns:**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 326).

**6.106.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType () const** [inline, virtual]

Retrieve the Destination Type for this Destination.

**Returns:**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 327).

References **cms::Destination::TOPIC**.

**6.106.2.10**    **virtual std::string activemq::commands::ActiveMQTopic::getTopicName  
() const throw ( cms::CMSExcption ) [inline, virtual]**

Gets the name of this topic.

**Returns:**

The topic name.

Implements **cms::Topic** (p. 3817).

**6.106.2.11**    **virtual std::string activemq::commands::ActiveMQTopic::toString ()  
const [virtual]**

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 330).

### 6.106.3 Field Documentation

**6.106.3.1**    **const unsigned char activemq::commands::ActiveMQTopic::ID\_-  
ACTIVEMQTOPIC = 101 [static]**

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

## 6.107 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 695).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.107.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 695).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.107.2 Constructor & Destructor Documentation

**6.107.2.1** `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.107.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.107.3 Member Function Documentation

**6.107.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.107.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.107.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 334).

**6.107.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 334).

**6.107.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 335).

**6.107.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 335).

**6.107.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 336).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h



## 6.108 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p.699).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.108.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p.699).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.108.2 Constructor & Destructor Documentation

**6.108.2.1** `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.108.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.108.3 Member Function Documentation

**6.108.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.108.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.108.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 342).

**6.108.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 342).

**6.108.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 343).

**6.108.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 343).

**6.108.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 344).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h`

## 6.109 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 703).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.109.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 703).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.109.2 Constructor & Destructor Documentation

**6.109.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.109.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.109.3 Member Function Documentation

**6.109.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.109.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.109.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 338).

**6.109.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 338).

**6.109.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 339).

**6.109.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 339).

**6.109.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 340).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTopicMarshaller.h**



## 6.110 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 707).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.110.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 707).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.110.2 Constructor & Destructor Documentation

**6.110.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.110.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.110.3 Member Function Documentation

**6.110.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.110.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.110.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 346).

**6.110.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 346).

**6.110.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 347).

**6.110.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 347).

**6.110.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 348).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h`

## 6.111 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 711).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.111.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 711).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.111.2 Constructor & Destructor Documentation

**6.111.2.1** `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.111.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.111.3 Member Function Documentation

**6.111.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.111.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.111.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 354).

**6.111.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 354).

**6.111.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 355).

**6.111.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 355).

**6.111.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 356).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h`



## 6.112 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 715).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.112.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 715).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.112.2 Constructor & Destructor Documentation

**6.112.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

**6.112.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

## 6.112.3 Member Function Documentation

**6.112.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.112.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.112.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 350).

**6.112.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 350).

**6.112.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 351).

**6.112.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 351).

**6.112.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 352).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h`

## 6.113 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

### Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** \*session, const **decaf::util::Properties** &properties)  
*Constructor.*
- virtual **~ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)  
*Adds a **Synchronization** (p. 3715) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)  
*Removes a **Synchronization** (p. 3715) to this Transaction.*
- virtual void **begin** () throw ( exceptions::ActiveMQException )  
*Begins a new transaction if one is not currently in progress.*
- virtual void **commit** () throw ( exceptions::ActiveMQException )  
*Commit the current Transaction.*
- virtual void **rollback** () throw ( exceptions::ActiveMQException )  
*Rollback the current Transaction.*
- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > & **getTransactionId** () const  
*Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.*
- virtual bool **isInTransaction** () const  
*Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.*

### 6.113.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

**Since:**

2.0

### 6.113.2 Constructor & Destructor Documentation

#### 6.113.2.1 `activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (ActiveMQSession * session, const decaf::util::Properties & properties)`

Constructor.

**Parameters:**

*session* The session that contains this transaction  
*properties* Configuration parameters for this object

#### 6.113.2.2 `virtual activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext ( ) [virtual]`

### 6.113.3 Member Function Documentation

#### 6.113.3.1 `virtual void activemq::core::ActiveMQTransactionContext::addSynchronization (const Pointer< Synchronization > & sync) [virtual]`

Adds a **Synchronization** (p. 3715) to this Transaction.

**Parameters:**

*sync* - The **Synchronization** (p. 3715) instance to add.

#### 6.113.3.2 `virtual void activemq::core::ActiveMQTransactionContext::begin () throw ( exceptions::ActiveMQException ) [virtual]`

Begins a new transaction if one is not currently in progress.

**Exceptions:**

*ActiveMQException*

#### 6.113.3.3 `virtual void activemq::core::ActiveMQTransactionContext::commit () throw ( exceptions::ActiveMQException ) [virtual]`

Commit the current Transaction.

**Exceptions:**

*ActiveMQException*

#### 6.113.3.4 `virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId () const [virtual]`

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

**Returns:**

TransactionInfo

**Exceptions:**

*InvalidStateException* if a Transaction is not in progress.

**6.113.3.5** virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const [virtual]

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

**Returns:**

true if a transaction is in progress.

**6.113.3.6** virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & sync) [virtual]

Removes a **Synchronization** (p. 3715) to this Transaction.

**Parameters:**

*sync* - The **Synchronization** (p. 3715) instance to add.

**6.113.3.7** virtual void activemq::core::ActiveMQTransactionContext::rollback () throw ( exceptions::ActiveMQException ) [virtual]

Rollback the current Transaction.

**Exceptions:**

*ActiveMQException*

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQTransactionContext.h

## 6.114 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 1142) for a data stream.

```
#include <src/main/decaf/util/zip/Adler32.h>Inheritance diagram for de-
caf::util::zip::Adler32:
```

### Public Member Functions

- **Adler32** ()
- virtual **~Adler32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()
 

*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)
 

*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
 

*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
 

*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)
 

*Updates the current checksum with the specified byte value.*

### 6.114.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p. 1142) for a data stream. The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

**Since:**

1.0



## 6.114.2 Constructor & Destructor Documentation

**6.114.2.1** decaf::util::zip::Adler32::Adler32 ()

**6.114.2.2** virtual decaf::util::zip::Adler32::~~Adler32 () [virtual]

## 6.114.3 Member Function Documentation

**6.114.3.1** virtual long long decaf::util::zip::Adler32::getValue () const [virtual]

### Returns:

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1142).

**6.114.3.2** virtual void decaf::util::zip::Adler32::reset () [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.114.3.3** virtual void decaf::util::zip::Adler32::update (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

### Parameters:

*byte* The byte value to update the current **Checksum** (p. 1142) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.114.3.4** virtual void decaf::util::zip::Adler32::update (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Updates the current checksum with the specified array of bytes.

### Parameters:

*buffer* The buffer to read the updated bytes from.

*size* The size of the passed buffer.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

### Exceptions:

**NullPointerException** if the passed buffer is NULL.

**IndexOutOfBoundsException** if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.114.3.5** `virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Updates the current checksum with the specified array of bytes.

**Parameters:**

*buffer* The buffer to read the updated bytes from.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

**Exceptions:**

*IndexOutOfBoundsException* if  $\text{offset} + \text{length} > \text{size of the buffer}$ .

Implements `decaf::util::zip::Checksum` (p. 1143).

**6.114.3.6** `virtual void decaf::util::zip::Adler32::update (const std::vector< unsigned char > & buffer)` [virtual]

Updates the current checksum with the specified vector of bytes.

**Parameters:**

*buffer* The buffer to read the updated bytes from.

Implements `decaf::util::zip::Checksum` (p. 1144).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Adler32.h`

## 6.115 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

`#include <src/main/decaf/lang/Appendable.h>` Inheritance diagram for decaf::lang::Appendable:

### Public Member Functions

- virtual `~Appendable()`
- virtual `Appendable & append(char value)=0` throw ( decaf::lang::Exception )

*Appends the specified character to this **Appendable** (p. 725).*

- virtual `Appendable & append(const CharSequence *csq)=0` throw ( decaf::lang::Exception )

*Appends the specified character sequence to this **Appendable** (p. 725).*

- virtual `Appendable & append(const CharSequence *csq, int start, int end)=0` throw ( decaf::lang::Exception )

*Appends a subsequence of the specified character sequence to this **Appendable** (p. 725).*

### 6.115.1 Detailed Description

An object to which char sequences and values can be appended. The **Appendable** (p.725) interface must be implemented by any class whose instances are intended to receive formatted output from a `Formatter`.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p.1100) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p.3765) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

**Since:**

1.0

## 6.115.2 Constructor & Destructor Documentation

6.115.2.1 `virtual decaf::lang::Appendable::~~Appendable () [inline, virtual]`

## 6.115.3 Member Function Documentation

6.115.3.1 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq, int start, int end) throw ( decaf::lang::Exception ) [pure virtual]`

Appends a subsequence of the specified character sequence to this **Appendable** (p. 725).

### Parameters:

*csq* - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

*start* The index of the first character in the subsequence.

*end* The index of the character following the last character in the subsequence.

### Returns:

a Reference to this **Appendable** (p. 725)

### Exceptions:

*Exception* (p. 1831) if an error occurs.

*IndexOutOfBoundsException* *start* is greater than *end*, or *end* is greater than *csq.length()*

Implemented in **decaf::io::Writer** (p. 4022), and **decaf::nio::CharBuffer** (p. 1123).

6.115.3.2 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq) throw ( decaf::lang::Exception ) [pure virtual]`

Appends the specified character sequence to this **Appendable** (p. 725).

### Parameters:

*csq* The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

### Returns:

a Reference to this **Appendable** (p. 725).

### Exceptions:

*Exception* (p. 1831) if an error occurs.

Implemented in **decaf::io::Writer** (p. 4023), and **decaf::nio::CharBuffer** (p. 1123).

6.115.3.3 `virtual Appendable& decaf::lang::Appendable::append (char value) throw ( decaf::lang::Exception ) [pure virtual]`

Appends the specified character to this **Appendable** (p. 725).

**Parameters:**

*value* The character to append.

**Returns:**

a Reference to this **Appendable** (p. 725)

**Exceptions:**

*Exception* (p. 1831) if an error occurs.

Implemented in **decaf::io::Writer** (p. 4023), and **decaf::nio::CharBuffer** (p. 1124).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Appendable.h`

## 6.116 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in **decaf** (p.143) can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

### Public Member Functions

- **AprPool** ()
- virtual **~AprPool** ()
- **apr\_pool\_t \* getAprPool** () const  
*Gets the **internal** (p. 144) APR Pool.*
- void **cleanup** ()  
*Clears data that was allocated by this pool.*

### Static Public Member Functions

- static **apr\_pool\_t \* getGlobalPool** ()  
*Gets a pointer to the Global APR Pool used for the Application.*

#### 6.116.1 Detailed Description

Wraps an APR pool object so that classes in **decaf** (p.143) can create a static member for use in static methods where apr function calls that need a pool are made.

#### 6.116.2 Constructor & Destructor Documentation

**6.116.2.1 decaf::internal::AprPool::AprPool ()**

**6.116.2.2 virtual decaf::internal::AprPool::~~AprPool ()** [virtual]

#### 6.116.3 Member Function Documentation

**6.116.3.1 void decaf::internal::AprPool::cleanup ()**

Clears data that was allocated by this pool. Users should call this after getting the data from the APR functions and copying it to someplace safe.

**6.116.3.2 apr\_pool\_t\* decaf::internal::AprPool::getAprPool () const**

Gets the **internal** (p.144) APR Pool.

#### Returns:

the **internal** (p.144) APR pool

**6.116.3.3 static apr\_pool\_t\* decaf::internal::AprPool::getGlobalPool () [static]**

Gets a pointer to the Global APR Pool used for the Application.

**Returns:**

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**AprPool.h**

## 6.117 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

### Data Structures

- struct **ArrayData**

### Public Types

- typedef T \* **PointerType**
- typedef T & **ReferenceType**
- typedef const T & **ConstReferenceType**
- typedef REFCOUNTER **CounterType**

### Public Member Functions

- **ArrayPointer** ()  
*Default Constructor.*
- **ArrayPointer** (int size)  
*Create a new **ArrayPointer** (p. 730) instance and allocates an **internal** (p. 144) array that is sized using the passed in size value.*
- **ArrayPointer** (const **PointerType** value, int size)  
*Explicit Constructor, creates an **ArrayPointer** (p. 730) that contains value with a single reference.*
- **ArrayPointer** (const **ArrayPointer** &value) throw ()  
*Copy constructor.*
- virtual ~**ArrayPointer** () throw ()
- void **reset** (T \*value, int size=0)  
*Resets the **ArrayPointer** (p. 730) to hold the new value.*
- T \* **release** ()  
*Releases the **Pointer** (p. 2946) held and resets the **internal** (p. 144) pointer value to Null.*
- **PointerType** **get** () const  
*Gets the real array pointer that is contained within this **Pointer** (p. 2946).*
- int **length** () const  
*Returns the current size of the contained array or zero if the array is NULL.*
- void **swap** (**ArrayPointer** &value) throw ()



*Exception* (p. 1831) Safe Swap Function.

- **ArrayPointer** clone () const

*Creates a new **ArrayPointer** (p. 730) instance that is a clone of the value contained in this **ArrayPointer** (p. 730).*

- **ArrayPointer** & operator= (const **ArrayPointer** &right) throw ()

*Assigns the value of right to this **Pointer** (p. 2946) and increments the reference Count.*

- template<typename T1 , typename R1 >

**ArrayPointer** & operator= (const **ArrayPointer**< T1, R1 > &right) throw ()

- **ReferenceType** operator[] (int index)

*Dereference Operator, returns a reference to the Contained value.*

- **ConstReferenceType** operator[] (int index) const

- bool operator! () const

- template<typename T1 , typename R1 >

bool operator== (const **ArrayPointer**< T1, R1 > &right) const

- template<typename T1 , typename R1 >

bool operator!= (const **ArrayPointer**< T1, R1 > &right) const

## Friends

- bool operator== (const **ArrayPointer** &left, const T \*right)

- bool operator== (const T \*left, const **ArrayPointer** &right)

- bool operator!= (const **ArrayPointer** &left, const T \*right)

- bool operator!= (const T \*left, const **ArrayPointer** &right)

### 6.117.1 Detailed Description

template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> class decaf::lang::ArrayPointer<  
T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used. This **Pointer** (p. 2946) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2946) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2946) in a STL container that requires it, **Pointer** (p. 2946) provides an implementation of std::less.

Since:

1.0

## 6.117.2 Member Typedef Documentation

**6.117.2.1** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef const T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType`

**6.117.2.2** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef REFCOUNTER decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType`

**6.117.2.3** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T* decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType`

**6.117.2.4** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType`

## 6.117.3 Constructor & Destructor Documentation

**6.117.3.1** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer () [inline]`

Default Constructor. Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, and `decaf::lang::ArrayPointer< unsigned char >::reset()`.

**6.117.3.2** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (int size) [inline]`

Create a new **ArrayPointer** (p. 730) instance and allocates an **internal** (p. 144) array that is sized using the passed in size value.

### Parameters:

*size* The size of the array to allocate for this **ArrayPointer** (p. 730) instance.

**6.117.3.3** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (const PointerType value, int size) [inline, explicit]`

Explicit Constructor, creates an **ArrayPointer** (p. 730) that contains value with a single reference. This object now has ownership until a call to release.

**Parameters:**

*value* The pointer to the instance of the array we are taking ownership of.

*size* The size of the array this object is taking ownership of.

```
6.117.3.4  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount>
           decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer (const
           ArrayPointer< T, REFCOUNTER > & value) throw () [inline]
```

Copy constructor. Copies the value contained in the **ArrayPointer** (p. 730) to the new instance and increments the reference counter.

```
6.117.3.5  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> virtual
           decaf::lang::ArrayPointer< T, REFCOUNTER >::~~ArrayPointer ()
           throw () [inline, virtual]
```

**6.117.4 Member Function Documentation**

```
6.117.4.1  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ArrayPointer
           decaf::lang::ArrayPointer< T, REFCOUNTER >::clone () const
           [inline]
```

Creates a new **ArrayPointer** (p. 730) instance that is a clone of the value contained in this **ArrayPointer** (p. 730).

**Returns:**

an **ArrayPointer** (p. 730) that contains a copy of the data in this **ArrayPointer** (p. 730).

```
6.117.4.2  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> PointerType
           decaf::lang::ArrayPointer< T, REFCOUNTER >::get () const [inline]
```

Gets the real array pointer that is contained within this **Pointer** (p. 2946). This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2946). Use at your own risk.

**Returns:**

the contained pointer.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, `decaf::lang::ArrayPointerComparator< T, R >::compare()`, `decaf::lang::operator!=()`, `decaf::lang::ArrayPointer< unsigned char >::operator!=()`, `std::less< decaf::lang::ArrayPointer< T > >::operator()`, `decaf::lang::ArrayPointerComparator< T, R >::operator()`, `decaf::lang::operator==()`, and `decaf::lang::ArrayPointer< unsigned char >::operator==()`.

**6.117.4.3** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> int  
decaf::lang::ArrayPointer< T, REFCOUNTER >::length () const  
[inline]`

Returns the current size of the contained array or zero if the array is NULL.

**Returns:**

the size of the array or zero if the array is NULL

**6.117.4.4** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! () const  
[inline]`

**6.117.4.5** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename  
T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER  
>::operator!= (const ArrayPointer< T1, R1 > & right) const [inline]`

**6.117.4.6** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename  
T1 , typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T,  
REFCOUNTER >::operator= (const ArrayPointer< T1, R1 > & right)  
throw () [inline]`

**6.117.4.7** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer&  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= (const  
ArrayPointer< T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 2946) and increments the reference Count.

**Parameters:**

*right* - **Pointer** (p. 2946) on the right hand side of an operator= call to this.

**6.117.4.8** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 , typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER  
>::operator==(const ArrayPointer< T1, R1 > & right) const [inline]`

**6.117.4.9** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> ConstReferenceType  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int index)  
const [inline]`

**6.117.4.10** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] (int index)  
[inline]`

Dereference Operator, returns a reference to the Contained value. This method throws an `NullPointerException` if the contained value is `NULL`.

**Returns:**

reference to the contained pointer.

**Exceptions:**

***NullPointerException*** if the contained value is `Null`

**6.117.4.11** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> T*  
decaf::lang::ArrayPointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p. 2946) held and resets the **internal** (p. 144) pointer value to `Null`. This method is not guaranteed to be safe if the **Pointer** (p. 2946) is held by more than one object or this method is called from more than one thread.

**Parameters:**

***value*** - The new value to contain.

**Returns:**

The pointer instance that was held by this **Pointer** (p. 2946) object, the pointer is no longer owned by this **Pointer** (p. 2946) and won't be freed when this **Pointer** (p. 2946) goes out of scope.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::ArrayPointer()`.

**6.117.4.12** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> void  
decaf::lang::ArrayPointer< T, REFCOUNTER >::reset (T * value, int  
size = 0) [inline]`

Resets the **ArrayPointer** (p. 730) to hold the new value. Before the new value is stored reset checks if the old value should be destroyed and if so calls `delete`. Call reset with a value of `NULL` is supported and acts to set this **Pointer** (p. 2946) to a `NULL` pointer.

**Parameters:**

*value* The new array pointer value to contain.

*size* The size of the new array value this object now contains.

**6.117.4.13** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> void  
decaf::lang::ArrayPointer< T, REFCOUNTER >::swap (ArrayPointer<  
T, REFCOUNTER > & value) throw () [inline]`

**Exception** (p.1831) Safe Swap Function.

**Parameters:**

*value* - the value to swap with this.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::operator=()`, and  
`decaf::lang::ArrayPointer< unsigned char >::swap()`.

**6.117.5 Friends And Related Function Documentation**

**6.117.5.1** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!=  
(const T * left, const ArrayPointer< T, REFCOUNTER > & right)  
[friend]`

**6.117.5.2** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!=  
(const ArrayPointer< T, REFCOUNTER > & left, const T * right)  
[friend]`

**6.117.5.3** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator==  
(const T * left, const ArrayPointer< T, REFCOUNTER > & right)  
[friend]`

**6.117.5.4** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator==  
(const ArrayPointer< T, REFCOUNTER > & left, const T * right)  
[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

## 6.118 decaf::lang::ArrayPointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 730).

#include <src/main/decaf/lang/ArrayPointer.h>Inheritance diagram for decaf::lang::ArrayPointerComparator< T, R >:

### Public Member Functions

- virtual bool **operator()** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const
- virtual int **compare** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const

### 6.118.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>
class decaf::lang::ArrayPointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 730). This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

### 6.118.2 Member Function Documentation

**6.118.2.1** template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual int decaf::lang::ArrayPointerComparator< T, R >::compare (const **ArrayPointer**< T, R > & *left*, const **ArrayPointer**< T, R > & *right*) const [inline, virtual]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

**6.118.2.2** template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool decaf::lang::ArrayPointerComparator< T, R >::operator() (const **ArrayPointer**< T, R > & *left*, const **ArrayPointer**< T, R > & *right*) const [inline, virtual]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`



## 6.119 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

### Public Member Functions

- **AtomicBoolean** ()  
*Creates a new **AtomicBoolean** (p. 738) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)  
*Creates a new **AtomicBoolean** (p. 738) with the initial value.*
- virtual ~**AtomicBoolean** ()
- bool **get** () const  
*Gets the current value of this **AtomicBoolean** (p. 738).*
- void **set** (bool newValue)  
*Unconditionally sets to the given value.*
- bool **compareAndSet** (bool expect, bool update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- bool **getAndSet** (bool newValue)  
*Atomically sets to the given value and returns the previous value.*
- std::string **toString** () const  
*Returns the String representation of the current value.*

### 6.119.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 738) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

### 6.119.2 Constructor & Destructor Documentation

#### 6.119.2.1 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()

Creates a new **AtomicBoolean** (p. 738) whose initial value is false.

#### 6.119.2.2 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool initialValue)

Creates a new **AtomicBoolean** (p. 738) with the initial value.

#### Parameters:

*initialValue* - The initial value of this boolean.

**6.119.2.3**    `virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean  
() [inline, virtual]`

### 6.119.3    Member Function Documentation

**6.119.3.1**    `bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet  
(bool expect, bool update)`

Atomically sets the value to the given updated value if the current value == the expected value.

**Parameters:**

*expect* - the expected value

*update* - the new value

**Returns:**

true if successful. False return indicates that the actual value was not equal to the expected value.

**6.119.3.2**    `bool decaf::util::concurrent::atomic::AtomicBoolean::get () const  
[inline]`

Gets the current value of this **AtomicBoolean** (p. 738).

**Returns:**

the currently set value.

**6.119.3.3**    `bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool  
new Value)`

Atomically sets to the given value and returns the previous value.

**Parameters:**

*new Value* - the new value

**Returns:**

the previous value

**6.119.3.4**    `void decaf::util::concurrent::atomic::AtomicBoolean::set (bool new Value)  
[inline]`

Unconditionally sets to the given value.

**Parameters:**

*new Value* - the new value

### 6.119.3.5 std::string decaf::util::concurrent::atomic::AtomicBoolean::toString () const

Returns the String representation of the current value.

**Returns:**

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicBoolean.h**

## 6.120 `decaf::util::concurrent::atomic::AtomicInteger` Class Reference

An int value that may be updated atomically.

`#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>` Inheritance diagram for `decaf::util::concurrent::atomic::AtomicInteger`:

### Public Member Functions

- **AtomicInteger** ()  
*Create a new **AtomicInteger** (p. 741) with an initial value of 0.*
- **AtomicInteger** (int initialValue)  
*Create a new **AtomicInteger** (p. 741) with the given initial value.*
- virtual **~AtomicInteger** ()
- int **get** () const  
*Gets the current value.*
- void **set** (int newValue)  
*Sets to the given value.*
- int **getAndSet** (int newValue)  
*Atomically sets to the given value and returns the old value.*
- bool **compareAndSet** (int expect, int update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- int **getAndIncrement** ()  
*Atomically increments by one the current value.*
- int **getAndDecrement** ()  
*Atomically decrements by one the current value.*
- int **getAndAdd** (int delta)  
*Atomically adds the given value to the current value.*
- int **incrementAndGet** ()  
*Atomically increments by one the current value.*
- int **decrementAndGet** ()  
*Atomically decrements by one the current value.*
- int **addAndGet** (int delta)  
*Atomically adds the given value to the current value.*
- std::string **toString** () const

*Returns the String representation of the current value.*

- `int intValue () const`

*Description copied from class: Number Returns the value of the specified number as an int.*

- `long long longValue () const`

*Description copied from class: Number Returns the value of the specified number as a long.*

- `float floatValue () const`

*Description copied from class: Number Returns the value of the specified number as a float.*

- `double doubleValue () const`

*Description copied from class: Number Returns the value of the specified number as a double.*

### 6.120.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 741) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an `Integer`. However, this class does extend `Number` to allow uniform access by tools and utilities that deal with numerically-based classes.

### 6.120.2 Constructor & Destructor Documentation

#### 6.120.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 741) with an initial value of 0.

#### 6.120.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 741) with the given initial value.

##### Parameters:

*initialValue* - The initial value of this object.

#### 6.120.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger () [inline, virtual]

### 6.120.3 Member Function Documentation

#### 6.120.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

##### Parameters:

*delta* - the value to add.

**Returns:**

the updated value.

**6.120.3.2    `bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int expect, int update)`**

Atomically sets the value to the given updated value if the current value == the expected value.

**Parameters:**

*expect* - the expected value

*update* - the new value

**Returns:**

true if successful. False return indicates that the actual value was not equal to the expected value.

**6.120.3.3    `int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()`**

Atomically decrements by one the current value.

**Returns:**

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::release()`.

**6.120.3.4    `double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]`**

Description copied from class: Number Returns the value of the specified number as a double. This may involve rounding.

**Returns:**

the numeric value represented by this object after conversion to type double.

Implements `decaf::lang::Number` (p. 2836).

**6.120.3.5    `float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]`**

Description copied from class: Number Returns the value of the specified number as a float. This may involve rounding.

**Returns:**

the numeric value represented by this object after conversion to type float.

Implements `decaf::lang::Number` (p. 2836).

**6.120.3.6** `int decaf::util::concurrent::atomic::AtomicInteger::get () const [inline]`

Gets the current value.

**Returns:**

the current value.

**6.120.3.7** `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int delta)`

Atomically adds the given value to the current value.

**Parameters:**

*delta* - The value to add.

**Returns:**

the previous value.

**6.120.3.8** `int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ()`

Atomically decrements by one the current value.

**Returns:**

the previous value.

**6.120.3.9** `int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()`

Atomically increments by one the current value.

**Returns:**

the previous value.

**6.120.3.10** `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int new Value)`

Atomically sets to the given value and returns the old value.

**Parameters:**

*new Value* - the new value.

**Returns:**

the previous value.

**6.120.3.11** `int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()`

Atomically increments by one the current value.

**Returns:**

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter()`.

**6.120.3.12** `int decaf::util::concurrent::atomic::AtomicInteger::intValue () const`  
`[virtual]`

Description copied from class: `Number` Returns the value of the specified number as an int. This may involve rounding or truncation.

**Returns:**

the numeric value represented by this object after conversion to type int.

Implements `decaf::lang::Number` (p. 2836).

**6.120.3.13** `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ()`  
`const [virtual]`

Description copied from class: `Number` Returns the value of the specified number as a long. This may involve rounding or truncation.

**Returns:**

the numeric value represented by this object after conversion to type long long.

Implements `decaf::lang::Number` (p. 2836).

**6.120.3.14** `void decaf::util::concurrent::atomic::AtomicInteger::set (int new Value)`  
`[inline]`

Sets to the given value.

**Parameters:**

*new Value* - the new value

**6.120.3.15** `std::string decaf::util::concurrent::atomic::AtomicInteger::toString ()`  
`const`

Returns the String representation of the current value.

**Returns:**

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicInteger.h`



## 6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference

```
#include <src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h>
```

Inherited by `decaf::lang::ArrayPointer< unsigned char >`, `decaf::lang::Pointer< activemq::threads::TaskRunner >`, `decaf::lang::Pointer< ActiveMQDestination >`, `decaf::lang::Pointer< ActiveMQTransactionContext >`, `decaf::lang::Pointer< BackupTransportPool >`, `decaf::lang::Pointer< BooleanExpression >`, `decaf::lang::Pointer< BrokerError >`, `decaf::lang::Pointer< BrokerId >`, `decaf::lang::Pointer< ByteArrayAdapter >`, `decaf::lang::Pointer< CloseTransportsTask >`, `decaf::lang::Pointer< cms::Destination >`, `decaf::lang::Pointer< commands::ActiveMQDestination >`, `decaf::lang::Pointer< commands::ConsumerId >`, `decaf::lang::Pointer< commands::ConsumerInfo >`, `decaf::lang::Pointer< commands::Message >`, `decaf::lang::Pointer< commands::MessageAck >`, `decaf::lang::Pointer< commands::ProducerInfo >`, `decaf::lang::Pointer< commands::SessionInfo >`, `decaf::lang::Pointer< commands::TransactionId >`, `decaf::lang::Pointer< commands::WireFormatInfo >`, `decaf::lang::Pointer< Comparator< E > >`, `decaf::lang::Pointer< CompositeTaskRunner >`, `decaf::lang::Pointer< ConnectionId >`, `decaf::lang::Pointer< ConnectionInfo >`, `decaf::lang::Pointer< ConsumerId >`, `decaf::lang::Pointer< ConsumerInfo >`, `decaf::lang::Pointer< core::ActiveMQAckHandler >`, `decaf::lang::Pointer< DataStructure >`, `decaf::lang::Pointer< decaf::lang::Runnable >`, `decaf::lang::Pointer< decaf::lang::Thread >`, `decaf::lang::Pointer< Exception >`, `decaf::lang::Pointer< LockHandle >`, `decaf::lang::Pointer< LogManagerInternals >`, `decaf::lang::Pointer< Message >`, `decaf::lang::Pointer< MessageAck >`, `decaf::lang::Pointer< MessageId >`, `decaf::lang::Pointer< ProducerId >`, `decaf::lang::Pointer< ProducerInfo >`, `decaf::lang::Pointer< Properties >`, `decaf::lang::Pointer< Response >`, `decaf::lang::Pointer< ResponseBuilder >`, `decaf::lang::Pointer< SessionId >`, `decaf::lang::Pointer< SessionInfo >`, `decaf::lang::Pointer< Tracked >`, `decaf::lang::Pointer< TransactionId >`, `decaf::lang::Pointer< TransactionState >`, `decaf::lang::Pointer< Transport >`, `decaf::lang::Pointer< TransportListener >`, `decaf::lang::Pointer< URI >`, `decaf::lang::Pointer< URIPool >`, and `decaf::lang::Pointer< wireformat::WireFormat >`.

### Public Member Functions

- `AtomicRefCounter ()`
- `AtomicRefCounter (const AtomicRefCounter &other)`
- `virtual ~AtomicRefCounter ()`

### Protected Member Functions

- `void swap (AtomicRefCounter &other)`

*Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.*

- `bool release ()`

*Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the **internal** (p. 144) counter is destroyed and this instance is now considered to be unreferenced.*

## 6.121.1 Constructor & Destructor Documentation

**6.121.1.1** `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter ()`  
[inline]

**6.121.1.2** `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter`  
`(const AtomicRefCounter & other)` [inline]

References `decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet()`.

**6.121.1.3** `virtual`  
`decaf::util::concurrent::atomic::AtomicRefCounter::~~AtomicRefCounter`  
`()` [inline, virtual]

## 6.121.2 Member Function Documentation

**6.121.2.1** `bool decaf::util::concurrent::atomic::AtomicRefCounter::release ()`  
[inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the **internal** (p. 144) counter is destroyed and this instance is now considered to be unreferenced.

### Returns:

true if the count is now zero.

Reimplemented in `decaf::lang::ArrayPointer< unsigned char >` (p. 735), `decaf::lang::Pointer< MessageAck >` (p. 2951), `decaf::lang::Pointer< BooleanExpression >` (p. 2951), `decaf::lang::Pointer< commands::ConsumerId >` (p. 2951), `decaf::lang::Pointer< BrokerError >` (p. 2951), `decaf::lang::Pointer< Transport >` (p. 2951), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 2951), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 2951), `decaf::lang::Pointer< CloseTransportsTask >` (p. 2951), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 2951), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 2951), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 2951), `decaf::lang::Pointer< Comparator< E > >` (p. 2951), `decaf::lang::Pointer< BrokerId >` (p. 2951), `decaf::lang::Pointer< LogManagerInternals >` (p. 2951), `decaf::lang::Pointer< commands::SessionInfo >` (p. 2951), `decaf::lang::Pointer< Message >` (p. 2951), `decaf::lang::Pointer< URI >` (p. 2951), `decaf::lang::Pointer< DataStructure >` (p. 2951), `decaf::lang::Pointer< activemq::threads::TaskRunner >` (p. 2951), `decaf::lang::Pointer< LockHandle >` (p. 2951), `decaf::lang::Pointer< commands::ActiveMQDestination >` (p. 2951), `decaf::lang::Pointer< ConsumerInfo >` (p. 2951), `decaf::lang::Pointer< ConnectionId >` (p. 2951), `decaf::lang::Pointer< decaf::lang::Runnable >` (p. 2951), `decaf::lang::Pointer< Properties >` (p. 2951), `decaf::lang::Pointer< BackupTransportPool >` (p. 2951), `decaf::lang::Pointer< ProducerInfo >` (p. 2951), `decaf::lang::Pointer< decaf::lang::Thread >` (p. 2951), `decaf::lang::Pointer< MessageId >` (p. 2951), `decaf::lang::Pointer< Response >` (p. 2951), `decaf::lang::Pointer< SessionId >` (p. 2951), `decaf::lang::Pointer< cms::Destination >` (p. 2951), `decaf::lang::Pointer< TransportListener >` (p. 2951), `decaf::lang::Pointer< commands::TransactionId >` (p. 2951), `decaf::lang::Pointer< ActiveMQDestination >` (p. 2951), `decaf::lang::Pointer< ProducerId >` (p. 2951), `decaf::lang::Pointer< ResponseBuilder >` (p. 2951), `decaf::lang::Pointer< SessionInfo >` (p. 2951), `decaf::lang::Pointer< commands::Message >` (p. 2951), `decaf::lang::Pointer<`

Tracked > (p. 2951), decaf::lang::Pointer< ConnectionInfo > (p. 2951), decaf::lang::Pointer< commands::MessageAck > (p. 2951), decaf::lang::Pointer< core::ActiveMQAckHandler > (p. 2951), decaf::lang::Pointer< Exception > (p. 2951), decaf::lang::Pointer< TransactionState > (p. 2951), decaf::lang::Pointer< commands::ConsumerInfo > (p. 2951), decaf::lang::Pointer< ConsumerId > (p. 2951), decaf::lang::Pointer< URIPool > (p. 2951), decaf::lang::Pointer< ByteArrayAdapter > (p. 2951), and decaf::lang::Pointer< TransactionId > (p. 2951).

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

#### 6.121.2.2 void decaf::util::concurrent::atomic::AtomicRefCounter::swap(AtomicRefCounter & *other*) [inline, protected]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

##### Parameters:

*other* The value to swap with this one's.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h

## 6.122    `decaf::util::concurrent::atomic::AtomicReference< T >`    Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

### Public Member Functions

- `AtomicReference ()`
- `AtomicReference (T *value)`
- `virtual ~AtomicReference ()`
- `T * get () const`

*Gets the Current Value.*

- `void set (T *newValue)`

*Sets the Current value of this Reference.*

- `bool compareAndSet (T *expect, T *update)`

*Atomically sets the value to the given updated value if the current value == the expected value.*

- `T * getAndSet (T *newValue)`

*Atomically sets to the given value and returns the old value.*

- `std::string toString () const`

*Returns the String representation of the current value.*

### 6.122.1 Detailed Description

```
template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >
```

An Pointer reference that may be updated atomically.

## 6.122.2 Constructor & Destructor Documentation

**6.122.2.1** `template<typename T >  
decaf::util::concurrent::atomic::AtomicReference< T  
>::AtomicReference () [inline]`

**6.122.2.2** `template<typename T >  
decaf::util::concurrent::atomic::AtomicReference< T  
>::AtomicReference (T * value) [inline]`

**6.122.2.3** `template<typename T > virtual  
decaf::util::concurrent::atomic::AtomicReference< T  
>::~~AtomicReference () [inline, virtual]`

## 6.122.3 Member Function Documentation

**6.122.3.1** `template<typename T > bool  
decaf::util::concurrent::atomic::AtomicReference< T  
>::compareAndSet (T * expect, T * update) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

### Parameters:

*expect* - the expected value

*update* - the new value

### Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

**6.122.3.2** `template<typename T > T*  
decaf::util::concurrent::atomic::AtomicReference< T >::get  
() const [inline]`

Gets the Current Value.

### Returns:

the current value of this Reference.

**6.122.3.3** `template<typename T > T*  
decaf::util::concurrent::atomic::AtomicReference< T  
>::getAndSet (T * new Value) [inline]`

Atomically sets to the given value and returns the old value.

### Parameters:

*new Value*- the new value

### Returns:

the previous value.

**6.122.3.4** `template<typename T > void  
decaf::util::concurrent::atomic::AtomicReference< T >::set  
(T * new Value) [inline]`

Sets the Current value of this Reference.

**Parameters:**

*new Value* The new Value of this Reference.

**6.122.3.5** `template<typename T > std::string  
decaf::util::concurrent::atomic::AtomicReference< T  
>::toString () const [inline]`

Returns the String representation of the current value.

**Returns:**

string representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicReference.h`

## 6.123 activemq::transport::failover::BackupTransport Class Reference

#include <src/main/activemq/transport/failover/BackupTransport.h> Inheritance diagram for activemq::transport::failover::BackupTransport:

### Public Member Functions

- **BackupTransport** (**BackupTransportPool** \*failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const  
*Gets the URI assigned to this Backup.*
- void **setUri** (const **decaf::net::URI** &uri)  
*Sets the URI assigned to this **Transport** (p. 3883).*
- const **Pointer**< **Transport** > & **getTransport** ()  
*Gets the currently held **transport** (p. 97).*
- void **setTransport** (const **Pointer**< **Transport** > &transport)  
*Sets the held **transport** (p. 97), if not NULL then start to listen for **exceptions** (p. 92) from the held **transport** (p. 97).*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command **transport** (p. 97).*
- bool **isClosed** () const  
*Has the **Transport** (p. 3883) been shutdown and no longer usable.*
- void **setClosed** (bool value)  
*Sets the closed flag on this **Transport** (p. 3883).*

### 6.123.1 Constructor & Destructor Documentation

- 6.123.1.1 **activemq::transport::failover::BackupTransport::BackupTransport** (**BackupTransportPool** \* failover)
- 6.123.1.2 virtual **activemq::transport::failover::BackupTransport::~~BackupTransport** ()  
[virtual]

### 6.123.2 Member Function Documentation

- 6.123.2.1 const **Pointer**<**Transport**>& **activemq::transport::failover::BackupTransport::getTransport** () [inline]

Gets the currently held **transport** (p. 97).

**Returns:**

pointer to the held **transport** (p. 97) or NULL if not set.

#### 6.123.2.2 `decaf::net::URI activemq::transport::failover::BackupTransport::getUri () const` [inline]

Gets the URI assigned to this Backup.

**Returns:**

the assigned URI

#### 6.123.2.3 `bool activemq::transport::failover::BackupTransport::isClosed () const` [inline]

Has the **Transport** (p. 3883) been shutdown and no longer usable.

**Returns:**

true if the **Transport** (p. 3883)

#### 6.123.2.4 `virtual void activemq::transport::failover::BackupTransport::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command **transport** (p. 97). The **BackupTransport** (p. 752) closes its internal **Transport** (p. 3883) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

**Parameters:**

*ex* The exception that was passed to this listener to handle.

Implements **activemq::transport::TransportListener** (p. 3901).

#### 6.123.2.5 `void activemq::transport::failover::BackupTransport::setClosed (bool value) [inline]`

Sets the closed flag on this **Transport** (p. 3883).

**Parameters:**

*value* - true for closed.

#### 6.123.2.6 `void activemq::transport::failover::BackupTransport::setTransport (const Pointer< Transport > & transport) [inline]`

Sets the held **transport** (p. 97), if not NULL then start to listen for **exceptions** (p. 92) from the held **transport** (p. 97).

**Parameters:**

*transport* (p. 97) The **transport** (p. 97) to hold.



**6.123.2.7** void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & *uri*) [inline]

Sets the URI assigned to this **Transport** (p. 3883).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

## 6.124 activemq::transport::failover::BackupTransportPool Class Reference

#include <src/main/activemq/transport/failover/BackupTransportPool.h> Inheritance diagram for activemq::transport::failover::BackupTransportPool:

### Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const  
*Return true if we don't currently have enough Connected Transports.*
- **Pointer**< **BackupTransport** > **getBackup** ()  
*Get a Connected **Transport** (p. 3883) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*
- virtual bool **iterate** ()  
*Connect to a Backup Broker if we haven't already connected to the max number of Backups.*
- int **getBackupPoolSize** () const  
*Gets the Max number of Backups this Task will create.*
- void **setBackupPoolSize** (int size)  
*Sets the Max number of Backups this Task will create.*
- bool **isEnabled** () const  
*Gets if the backup **Transport** (p. 3883) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*
- void **setEnabled** (bool value)  
*Sets if this Backup **Transport** (p. 3883) Pool is enabled.*

### Friends

- class **BackupTransport**

## 6.124.1 Constructor & Destructor Documentation

**6.124.1.1** `activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

**6.124.1.2** `activemq::transport::failover::BackupTransportPool::BackupTransportPool (int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

**6.124.1.3** `virtual  
activemq::transport::failover::BackupTransportPool::~~BackupTransportPool  
( ) [virtual]`

## 6.124.2 Member Function Documentation

**6.124.2.1** `Pointer<BackupTransport> ac-  
tivemq::transport::failover::BackupTransportPool::getBackup  
( )`

Get a Connected **Transport** (p. 3883) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

### Returns:

Pointer to a Connected **Transport** (p. 3883) or NULL

**6.124.2.2** `int ac-  
tivemq::transport::failover::BackupTransportPool::getBackupPoolSize ( )  
const [inline]`

Gets the Max number of Backups this Task will create.

### Returns:

the max number of active BackupTransports that will be created.

**6.124.2.3** `bool activemq::transport::failover::BackupTransportPool::isEnabled ( )  
const [inline]`

Gets if the backup **Transport** (p. 3883) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

### Returns:

true if enable.

**6.124.2.4** `virtual bool activemq::transport::failover::BackupTransportPool::isPending () const` [virtual]

Return true if we don't currently have enough Connected Transports.

Implements `activemq::threads::CompositeTask` (p. 1221).

**6.124.2.5** `virtual bool activemq::transport::failover::BackupTransportPool::iterate ()` [virtual]

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements `activemq::threads::Task` (p. 3734).

**6.124.2.6** `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int size)` [inline]

Sets the Max number of Backups this Task will create.

**Parameters:**

*size* - the max number of active BackupTransports that will be created.

**6.124.2.7** `void activemq::transport::failover::BackupTransportPool::setEnabled (bool value)`

Sets if this Backup **Transport** (p. 3883) Pool is enabled. When not enabled no Backups are created and any that were are destroyed.

**Parameters:**

*value* - true to enable backup creation, false to disable.

## 6.124.3 Friends And Related Function Documentation

**6.124.3.1** `friend class BackupTransport` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

## 6.125 activemq::commands::BaseCommand Class Reference

#include <src/main/activemq/commands/BaseCommand.h> Inheritance diagram for activemq::commands::BaseCommand:

### Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)  
*Sets the **Command** (p. 1194) Id of this **Message** (p. 2516).*
- virtual int **getCommandId** () const  
*Gets the **Command** (p. 1194) Id of this **Message** (p. 2516).*
- virtual void **setResponseRequired** (const bool required)  
*Set if this **Message** (p. 2516) requires a **Response** (p. 3285).*
- virtual bool **isResponseRequired** () const  
*Is a **Response** (p. 3285) required for this **Command** (p. 1194).*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

## 6.125.1 Constructor & Destructor Documentation

**6.125.1.1** `activemq::commands::BaseCommand::BaseCommand () [inline]`

**6.125.1.2** `virtual activemq::commands::BaseCommand::~~BaseCommand () [inline, virtual]`

## 6.125.2 Member Function Documentation

**6.125.2.1** `virtual void activemq::commands::BaseCommand::copyDataStructure (const DataStructure * src) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns:

`src` - Source Object

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 205), `activemq::commands::ActiveMQBytesMessage` (p. 237), `activemq::commands::ActiveMQMapMessage` (p. 364), `activemq::commands::ActiveMQMessage` (p. 399), `activemq::commands::ActiveMQObjectMessage` (p. 444), `activemq::commands::ActiveMQStreamMessage` (p. 540), `activemq::commands::ActiveMQTextMessage` (p. 664), `activemq::commands::BrokerError` (p. 864), `activemq::commands::BrokerInfo` (p. 898), `activemq::commands::ConnectionControl` (p. 1268), `activemq::commands::ConnectionError` (p. 1297), `activemq::commands::ConnectionInfo` (p. 1357), `activemq::commands::ConsumerControl` (p. 1402), `activemq::commands::ConsumerInfo` (p. 1461), `activemq::commands::ControlCommand` (p. 1494), `activemq::commands::DataArrayResponse` (p. 1529), `activemq::commands::DataResponse` (p. 1584), `activemq::commands::DestinationInfo` (p. 1728), `activemq::commands::ExceptionResponse` (p. 1840), `activemq::commands::FlushCommand` (p. 1938), `activemq::commands::IntegerResponse` (p. 2092), `activemq::commands::KeepAliveInfo` (p. 2267), `activemq::commands::Message` (p. 2521), `activemq::commands::MessageAck` (p. 2560), `activemq::commands::MessageDispatch` (p. 2595), `activemq::commands::MessageDispatchNotification` (p. 2631), `activemq::commands::MessagePull` (p. 2740), `activemq::commands::ProducerAck` (p. 3037), `activemq::commands::ProducerInfo` (p. 3097), `activemq::commands::RemoveInfo` (p. 3195), `activemq::commands::RemoveSubscriptionInfo` (p. 3223), `activemq::commands::ReplayCommand` (p. 3252), `activemq::commands::Response` (p. 3286), `activemq::commands::SessionInfo` (p. 3408), `activemq::commands::ShutdownInfo` (p. 3471), `activemq::commands::TransactionInfo` (p. 3848), and `activemq::commands::WireFormatInfo` (p. 3985).

References `getCommandId()`, and `isResponseRequired()`.

**6.125.2.2** `virtual bool activemq::commands::BaseCommand::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 206), **activemq::commands::ActiveMQBytesMessage** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 429), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 541), **activemq::commands::ActiveMQTextMessage** (p. 664), **activemq::commands::BrokerInfo** (p. 898), **activemq::commands::ConnectionControl** (p. 1269), **activemq::commands::ConnectionError** (p. 1297), **activemq::commands::ConnectionInfo** (p. 1357), **activemq::commands::ConsumerControl** (p. 1402), **activemq::commands::ConsumerInfo** (p. 1461), **activemq::commands::ControlCommand** (p. 1494), **activemq::commands::DataArrayResponse** (p. 1529), **activemq::commands::DataResponse** (p. 1584), **activemq::commands::DestinationInfo** (p. 1728), **activemq::commands::ExceptionResponse** (p. 1840), **activemq::commands::FlushCommand** (p. 1938), **activemq::commands::IntegerResponse** (p. 2092), **activemq::commands::KeepAliveInfo** (p. 2267), **activemq::commands::Message** (p. 2521), **activemq::commands::MessageAck** (p. 2560), **activemq::commands::MessageDispatch** (p. 2595), **activemq::commands::MessageDispatchNotification** (p. 2631), **activemq::commands::MessagePull** (p. 2740), **activemq::commands::ProducerAck** (p. 3037), **activemq::commands::ProducerInfo** (p. 3097), **activemq::commands::RemoveInfo** (p. 3195), **activemq::commands::RemoveSubscriptionInfo** (p. 3223), **activemq::commands::ReplayCommand** (p. 3252), **activemq::commands::Response** (p. 3286), **activemq::commands::SessionInfo** (p. 3408), **activemq::commands::ShutdownInfo** (p. 3471), **activemq::commands::TransactionInfo** (p. 3848), **activemq::commands::WireFormatInfo** (p. 3985), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 429), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 429).

References **activemq::commands::BaseDataStructure::equals()**.

### 6.125.2.3 virtual int activemq::commands::BaseCommand::getCommandId () const [inline, virtual]

Gets the **Command** (p. 1194) Id of this **Message** (p. 2516).

**Returns:**

**Command** (p. 1194) Id

Implements **activemq::commands::Command** (p. 1195).

Referenced by **copyDataStructure()**.

**6.125.2.4** `virtual bool activemq::commands::BaseCommand::isBrokerInfo () const`  
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 1195).

Reimplemented in `activemq::commands::BrokerInfo` (p. 900).

**6.125.2.5** `virtual bool activemq::commands::BaseCommand::isConnectionInfo ()`  
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1195).

Reimplemented in `activemq::commands::ConnectionInfo` (p. 1358).

**6.125.2.6** `virtual bool activemq::commands::BaseCommand::isConsumerInfo ()`  
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1195).

Reimplemented in `activemq::commands::ConsumerInfo` (p. 1463).

**6.125.2.7** `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ()`  
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1195).

Reimplemented in `activemq::commands::KeepAliveInfo` (p. 2268).

**6.125.2.8** `virtual bool activemq::commands::BaseCommand::isMessage () const`  
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 1195).

Reimplemented in `activemq::commands::Message` (p. 2527).

**6.125.2.9** `virtual bool activemq::commands::BaseCommand::isMessageAck () const`  
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::MessageAck` (p. 2562).

**6.125.2.10** `virtual bool activemq::commands::BaseCommand::isMessageDispatch ()`  
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::MessageDispatch` (p. 2596).



**6.125.2.11** `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::MessageDispatchNotification` (p. 2632).

**6.125.2.12** `virtual bool activemq::commands::BaseCommand::isProducerAck () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::ProducerAck` (p. 3038).

**6.125.2.13** `virtual bool activemq::commands::BaseCommand::isProducerInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::ProducerInfo` (p. 3098).

**6.125.2.14** `virtual bool activemq::commands::BaseCommand::isRemoveInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::RemoveInfo` (p. 3196).

**6.125.2.15** `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::RemoveSubscriptionInfo` (p. 3224).

**6.125.2.16** `virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1196).

Reimplemented in `activemq::commands::Response` (p. 3287).

**6.125.2.17** `virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]`

Is a **Response** (p. 3285) required for this **Command** (p. 1194).

#### Returns:

true if a response is required.

Implements **activemq::commands::Command** (p. 1197).

Referenced by `copyDataStructure()`.

**6.125.2.18** `virtual bool activemq::commands::BaseCommand::isShutdownInfo ()`  
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1197).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 3472).

**6.125.2.19** `virtual bool activemq::commands::BaseCommand::isTransactionInfo ()`  
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1197).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3849).

**6.125.2.20** `virtual bool activemq::commands::BaseCommand::isWireFormatInfo ()`  
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1197).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 3989).

**6.125.2.21** `virtual void activemq::commands::BaseCommand::setCommandId (int`  
`id) [inline, virtual]`

Sets the **Command** (p. 1194) Id of this **Message** (p. 2516).

**Parameters:**

*id* **Command** (p. 1194) Id

Implements **activemq::commands::Command** (p. 1197).

**6.125.2.22** `virtual void activemq::commands::BaseCommand::setResponseRequired`  
`(const bool required) [inline, virtual]`

Set if this **Message** (p. 2516) requires a **Response** (p. 3285).

**Parameters:**

*required* true if response is required

Implements **activemq::commands::Command** (p. 1197).

**6.125.2.23** `virtual std::string activemq::commands::BaseCommand::toString ()`  
`const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 1198).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 208), **activemq::commands::ActiveMQBytesMessage** (p. 244), **activemq::commands::ActiveMQMapMessage** (p. 372), **activemq::commands::ActiveMQMessage** (p. 400), **activemq::commands::ActiveMQObjectMessage** (p. 445), **activemq::commands::ActiveMQStreamMessage** (p. 547), **activemq::commands::ActiveMQTextMessage** (p. 666), **activemq::commands::BrokerInfo** (p. 901), **activemq::commands::ConnectionControl** (p. 1270), **activemq::commands::ConnectionError** (p. 1298), **activemq::commands::ConnectionInfo** (p. 1359), **activemq::commands::ConsumerControl** (p. 1404), **activemq::commands::ConsumerInfo** (p. 1464), **activemq::commands::ControlCommand** (p. 1495), **activemq::commands::DataArrayResponse** (p. 1530), **activemq::commands::DataResponse** (p. 1585), **activemq::commands::DestinationInfo** (p. 1730), **activemq::commands::ExceptionResponse** (p. 1841), **activemq::commands::FlushCommand** (p. 1939), **activemq::commands::IntegerResponse** (p. 2093), **activemq::commands::KeepAliveInfo** (p. 2268), **activemq::commands::Message** (p. 2530), **activemq::commands::MessageAck** (p. 2563), **activemq::commands::MessageDispatch** (p. 2597), **activemq::commands::MessageDispatchNotification** (p. 2633), **activemq::commands::MessagePull** (p. 2742), **activemq::commands::ProducerAck** (p. 3038), **activemq::commands::ProducerInfo** (p. 3099), **activemq::commands::RemoveInfo** (p. 3196), **activemq::commands::RemoveSubscriptionInfo** (p. 3225), **activemq::commands::ReplayCommand** (p. 3253), **activemq::commands::Response** (p. 3287), **activemq::commands::SessionInfo** (p. 3409), **activemq::commands::ShutdownInfo** (p. 3472), **activemq::commands::TransactionInfo** (p. 3849), and **activemq::commands::WireFormatInfo** (p. 3991).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

## 6.126 activemq::wireformat::openwire::marshal::v3::BaseCommandMarsh Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 765).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.126.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 765).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.126.2 Constructor & Destructor Documentation

- 6.126.2.1** `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller()` `[inline]`
- 6.126.2.2** `virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller()` `[inline, virtual]`

## 6.126.3 Member Function Documentation

- 6.126.3.1** `virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

### Exceptions:

- IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 210), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 250), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 553), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 668), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 905), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1305), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1366), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1473), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1536), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1737), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1851), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1949), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2103), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2278), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2574), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2611), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`

(p. 2644), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2699),  
`activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2753),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3049),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3118),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3207), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
(p. 3232), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
(p. 3267), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311),  
`activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3424), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3490), and  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3860).

**6.126.3.2 virtual void `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal`**  
**(`OpenWireFormat * wireFormat`, `commands::DataStructure`  
`* dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`)** [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

***wireFormat*** - describes the wire format of the broker  
***dataStructure*** - Object to be marshaled  
***dataIn*** - `BinaryReader` that provides that data source

#### Exceptions:

***IOException*** if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 251), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 376), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 403), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 448), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 669), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 906),  
`activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1274),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1306),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1367),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1412),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1474), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1537),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1600),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1738), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1852),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1950),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2104),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2279),

activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2575), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2612), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2645), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2699), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2754), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3050), activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3119), activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3208), activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 3233), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 3268), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3312), activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3425), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3491), and activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3861).

**6.126.3.3** virtual int activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 211), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 554), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 669), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 906), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1274), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1306), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1367), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1412), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1503), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1537),

**activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1600),  
**activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1738), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1852),  
**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1950),  
**activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2104),  
**activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2279),  
**activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2575), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2612), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2645),  
**activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2700),  
**activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2754),  
**activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 3050),  
**activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3119),  
**activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3208), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3233),  
**activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3268), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3312),  
**activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3425), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3491), and  
**activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3861).

**6.126.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2**  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 211), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 554), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 669), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 906), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1274), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1306), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1367),



activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1412),  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1474),  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1503),  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1538),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1601),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1738),  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1853),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1950),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2105),  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2279),  
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2575),  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2612),  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
 (p. 2645),  
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2701),  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2754),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3050),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3119),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3208),  
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
 (p. 3233),  
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
 (p. 3268),  
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3313),  
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3425),  
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3491), and  
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3861).

**6.126.3.5 virtual void** `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
 (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
 (p. 252), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
 (p. 377), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
 (p. 404), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
 (p. 449), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
 (p. 555), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`

(p. 670), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 907), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1275), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1307), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1413), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1475), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1504), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1538), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1601), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1739), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1853), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1951), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2105), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2280), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2576), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2613), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2646), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2701), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2755), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3051), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3120), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3209), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3234), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3269), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3314), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3426), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3492), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3862).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

## 6.127 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 772).

#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.127.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 772).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.127.2 Constructor & Destructor Documentation

**6.127.2.1** `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

**6.127.2.2** `virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

## 6.127.3 Member Function Documentation

**6.127.3.1** `virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 218), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 258), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 383), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 561), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 672), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 909), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1309), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1370), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1477), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1540), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1603), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1741), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1859), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1953), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2107), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2282), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2619), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`

(p. 2648), [activemq::wireformat::openwire::marshal::v4::MessageMarshaller](#) (p. 2709), [activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller](#) (p. 2757), [activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller](#) (p. 3045), [activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller](#) (p. 3102), [activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller](#) (p. 3219), [activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller](#) (p. 3248), [activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller](#) (p. 3255), [activemq::wireformat::openwire::marshal::v4::ResponseMarshaller](#) (p. 3296), [activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller](#) (p. 3432), [activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller](#) (p. 3494), and [activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller](#) (p. 3868).

**6.127.3.2 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1631).

Reimplemented in [activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller](#) (p. 219), [activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller](#) (p. 259), [activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller](#) (p. 384), [activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller](#) (p. 411), [activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller](#) (p. 456), [activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller](#) (p. 562), [activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller](#) (p. 673), [activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller](#) (p. 910), [activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller](#) (p. 1278), [activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller](#) (p. 1310), [activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller](#) (p. 1371), [activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller](#) (p. 1416), [activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller](#) (p. 1478), [activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller](#) (p. 1507), [activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller](#) (p. 1541), [activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller](#) (p. 1604), [activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller](#) (p. 1742), [activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller](#) (p. 1860), [activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller](#) (p. 1954), [activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller](#) (p. 2108), [activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller](#) (p. 2283),

`activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2579), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2620), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2649), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2709), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2758), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3046), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3103), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3220), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3249), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3256), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3433), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3495), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3869).

**6.127.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 259), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 384), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 411), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 562), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 673), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 910), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1310), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1371), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1478), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1507), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1541),

activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1604),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1742),  
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1860),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1954),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2108),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2283),  
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2579),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2620),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
 (p. 2649),  
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2710),  
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2758),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3046),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3103),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3220),  
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3249),  
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3256),  
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3297),  
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3433),  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3495),  
 and  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3869).

**6.127.3.4 virtual void** `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1645).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
 (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
 (p. 259), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
 (p. 384), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
 (p. 411), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
 (p. 456), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
 (p. 562), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
 (p. 673), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 910),  
`activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1278),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1371),

activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1416),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1478),  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1507),  
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1542),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1605),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1742),  
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1861),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1954),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2109),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2283),  
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2579),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2620),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
 (p. 2649), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2711),  
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2758),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3046),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3103),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3220),  
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3249), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3256), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3298),  
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3433),  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3495), and  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3869).

**6.127.3.5** virtual void ac-  
 tivemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
 (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
 (p. 260), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
 (p. 385), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
 (p. 412), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
 (p. 457), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
 (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`



(p. 674), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 911), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1279), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1311), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1417), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1479), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1508), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1542), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1605), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1743), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1861), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1955), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2109), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2284), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2580), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2621), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2650), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2711), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2759), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3047), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3104), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3221), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3250), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3257), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3299), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3434), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3496), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3870).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

## 6.128 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 779).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.128.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 779).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.128.2 Constructor & Destructor Documentation

- 6.128.2.1 `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]
- 6.128.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

## 6.128.3 Member Function Documentation

- 6.128.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

### Exceptions:

- IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 254), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 379), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 557), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 676), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 913), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1374), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1419), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1544), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1607), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1745), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1863), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1957), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2111), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2290), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2582), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2623), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`

(p. 2652), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2714), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2761), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3061), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3110), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3228), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3259), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3420), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3482), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3856).

**6.128.3.2** `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 380), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 407), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 558), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 677), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 914), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1282), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1375), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1420), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1482), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1511), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1545), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1608), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1746), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1864), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1958), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2112), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2291),

activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2583), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2624), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2653), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2714), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2762), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3062), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3111), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3212), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3229), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3260), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3317), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3421), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3483), and activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3857).

**6.128.3.3** virtual int activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 215), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 255), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 407), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 452), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 558), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 677), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 914), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1314), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1375), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1420), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1482), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1511), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1545),

**activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1608),  
**activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1746), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1864),  
**activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1958),  
**activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2112),  
**activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2291),  
**activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2583), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2624), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2653),  
**activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2715),  
**activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2762),  
**activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3062),  
**activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3111),  
**activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3212), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 3229),  
**activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3260), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3317),  
**activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3421), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3483), and  
**activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3857).

**6.128.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2**  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 215), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 255), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 407), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 452), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 558), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 677), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 914), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1314), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1375),

activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1420),  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1482),  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1511),  
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1546),  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1609),  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1746),  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1865),  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1958),  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2113),  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2291),  
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2583),  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2624),  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
 (p. 2653),  
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2716),  
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2762),  
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3062),  
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3111),  
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3212),  
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller  
 (p. 3229),  
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller  
 (p. 3260),  
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3318),  
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3421),  
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3483), and  
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3857).

**6.128.3.5** virtual void ac-  
 tivemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`  
 (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`  
 (p. 256), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`  
 (p. 381), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`  
 (p. 408), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`  
 (p. 453), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`  
 (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`

(p. 678), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 915), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1283), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1421), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1483), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1512), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1546), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1609), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1747), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1865), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1959), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2113), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2292), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2625), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2654), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2716), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2763), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3063), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3112), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3213), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3230), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3261), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3319), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3422), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3484), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3858).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h`



## 6.129 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 786).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.129.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 786).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.129.2 Constructor & Destructor Documentation

**6.129.2.1** `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

**6.129.2.2** `virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

## 6.129.3 Member Function Documentation

**6.129.3.1** `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 262), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 387), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 565), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 680), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 917), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1285), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1317), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1378), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1423), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1514), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1548), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1587), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1753), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1855), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1961), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2115), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2286), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2586), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2615), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`

(p. 2656), [activemq::wireformat::openwire::marshal::v5::MessageMarshaller](#) (p. 2694), [activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller](#) (p. 2749), [activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller](#) (p. 3053), [activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller](#) (p. 3114), [activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller](#) (p. 3215), [activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller](#) (p. 3244), [activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller](#) (p. 3275), [activemq::wireformat::openwire::marshal::v5::ResponseMarshaller](#) (p. 3306), [activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller](#) (p. 3416), [activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller](#) (p. 3486), and [activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller](#) (p. 3852).

**6.129.3.2 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1631).

Reimplemented in [activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller](#) (p. 227), [activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller](#) (p. 263), [activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller](#) (p. 388), [activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller](#) (p. 415), [activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller](#) (p. 460), [activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller](#) (p. 566), [activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller](#) (p. 681), [activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller](#) (p. 918), [activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller](#) (p. 1286), [activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller](#) (p. 1318), [activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller](#) (p. 1379), [activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller](#) (p. 1424), [activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller](#) (p. 1486), [activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller](#) (p. 1515), [activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller](#) (p. 1549), [activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller](#) (p. 1588), [activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller](#) (p. 1754), [activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller](#) (p. 1856), [activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller](#) (p. 1962), [activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller](#) (p. 2116), [activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller](#) (p. 2287),

**activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller** (p. 2587), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2616), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2657), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2694), **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2750), **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3054), **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3115), **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3216), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 3245), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3276), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3307), **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3417), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3487), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3853).

**6.129.3.3 virtual int activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1**  
**(OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 263), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 415), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 566), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 681), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 918), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1286), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1318), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1379), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1424), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1486), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1515), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1549),

activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1588),  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1754),  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1856),  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1962),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2116),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2287),  
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2587),  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2616),  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
 (p. 2657), activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2695),  
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2750),  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3054),  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3115),  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3216),  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
 (p. 3245), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
 (p. 3276), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3307),  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3417),  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3487), and  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3853).

**6.129.3.4 virtual void** `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1645).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
 (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
 (p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
 (p. 388), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
 (p. 415), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
 (p. 460), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
 (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
 (p. 681), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 918),  
`activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1286),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1318),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1379),

activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1424),  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1486),  
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1515),  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1550),  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1589),  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1754),  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1857),  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1962),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2117),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2287),  
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2587),  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2616),  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
 (p. 2657), activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2696),  
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2750),  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3054),  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3115),  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3216),  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
 (p. 3245), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
 (p. 3276), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3308),  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3417),  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3487), and  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3853).

**6.129.3.5** virtual void ac-  
 tivemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
 (p. 228), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
 (p. 264), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
 (p. 389), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
 (p. 416), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
 (p. 461), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
 (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`

(p. 682), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 919), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1287), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1319), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1425), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1487), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1516), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1550), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1755), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1857), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1963), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2117), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2288), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2588), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2617), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2658), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2696), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2751), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3055), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3116), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3217), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3246), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3277), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3309), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3418), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3488), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3854).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

## 6.130 activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 793).

#include <src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.130.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 793).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.130.2 Constructor & Destructor Documentation

**6.130.2.1** `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::BaseCommandMarshaller()` `[inline]`

**6.130.2.2** `virtual activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::~~BaseCommandMarshaller()` `[inline, virtual]`

## 6.130.3 Member Function Documentation

**6.130.3.1** `virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

### Exceptions:

- IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 266), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 467), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 573), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 684), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 921), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1382), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1427), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1489), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1552), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1749), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1843), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1941), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2095), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2270), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2566), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2627), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`

(p. 2636), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2765), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3057), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3122), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3240), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3271), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3412), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3474), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3864).

**6.130.3.2 virtual void `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseUnmarshal`**  
**(`OpenWireFormat * wireFormat`, `commands::DataStructure`  
`* dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`)** [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - `BinaryReader` that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 267), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 423), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 468), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 574), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 685), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 922), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1290), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1383), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1428), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1519), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1553), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1592), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1750), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1844), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1942), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2096), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2271),

activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller (p. 2567), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller (p. 2628), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller (p. 2637), activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2719), activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2766), activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3058), activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3123), activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3204), activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller (p. 3241), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller (p. 3272), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3322), activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3413), activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3475), and activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3865).

**6.130.3.3** virtual int activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 267), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 468), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 574), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 685), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 922), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1290), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1322), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1383), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1428), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1490), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1553),

**activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1592),  
**activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1750), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1844),  
**activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 1942),  
**activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2096),  
**activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2271),  
**activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2567), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2628), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller** (p. 2637), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2720),  
**activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2766),  
**activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3058),  
**activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3123),  
**activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3204), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3241), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3272), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3322),  
**activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3413), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3475), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3865).

**6.130.3.4 virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal2**  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 267), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 468), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 574), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 685), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 922), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1290), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1322), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1383),

activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1428),  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1490),  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1519),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1554),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1593),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1750),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1845),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1942),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2097),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2271),  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller (p. 2567),  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller (p. 2628),  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2637), activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2721),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2766),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3058),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3123),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3204),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3241), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3272), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3323),  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3413),  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3475), and  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3865).

**6.130.3.5 virtual void** `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightUnmarshal`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`  
 (p. 232), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`  
 (p. 268), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
 (p. 397), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
 (p. 424), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
 (p. 469), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
 (p. 575), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`

(p. 686), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 923), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1291), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1429), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1520), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1554), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1593), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1751), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1845), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1943), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2097), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2272), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2568), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2629), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2638), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2721), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2767), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3059), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3124), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3205), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3242), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3273), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3324), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3414), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3476), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3866).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h`

## 6.131 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 800).

#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.131.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 800).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.131.2 Constructor & Destructor Documentation

**6.131.2.1** `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

**6.131.2.2** `virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

## 6.131.3 Member Function Documentation

**6.131.3.1** `virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 270), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 391), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 569), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 688), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 925), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1293), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1301), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1362), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1469), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1532), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1733), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1847), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1945), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2099), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2274), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2570), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2607), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`



(p. 2640), [activemq::wireformat::openwire::marshal::v2::MessageMarshaller](#) (p. 2704), [activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller](#) (p. 2745), [activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller](#) (p. 3041), [activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller](#) (p. 3106), [activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller](#) (p. 3199), [activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller](#) (p. 3236), [activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller](#) (p. 3263), [activemq::wireformat::openwire::marshal::v2::ResponseMarshaller](#) (p. 3301), [activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller](#) (p. 3428), [activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller](#) (p. 3478), and [activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller](#) (p. 3872).

**6.131.3.2 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

#### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1631).

Reimplemented in [activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller](#) (p. 223), [activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller](#) (p. 271), [activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller](#) (p. 392), [activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller](#) (p. 419), [activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller](#) (p. 464), [activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller](#) (p. 570), [activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller](#) (p. 689), [activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller](#) (p. 926), [activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller](#) (p. 1294), [activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller](#) (p. 1302), [activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller](#) (p. 1363), [activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller](#) (p. 1408), [activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller](#) (p. 1470), [activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller](#) (p. 1499), [activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller](#) (p. 1533), [activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller](#) (p. 1596), [activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller](#) (p. 1734), [activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller](#) (p. 1848), [activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller](#) (p. 1946), [activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller](#) (p. 2100), [activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller](#) (p. 2275),

**activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2571), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2608), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 2641), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2704), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2746), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 3042), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 3107), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 3200), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 3237), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 3264), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3302), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 3429), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 3479), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 3873).

**6.131.3.3 virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1**  
**(OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 271), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 419), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 570), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 689), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 926), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1294), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1363), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1408), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1470), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1533),

activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1596),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1734),  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1848),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1946),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2100),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2275),  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2571),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2608),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2641), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2705),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2746),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3042),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3107),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3200),  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3237), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3264), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3302),  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3429),  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3479), and  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3873).

**6.131.3.4 virtual void** `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1645).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`  
 (p. 223), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
 (p. 271), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
 (p. 392), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
 (p. 419), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
 (p. 464), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
 (p. 570), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
 (p. 689), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 926),  
`activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1294),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1302),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1363),

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1408),  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1470),  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1499),  
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1534),  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1597),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1734),  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1849),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1946),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2101),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2275),  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2571),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2608),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2641), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2706),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2746),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3042),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3107),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3200),  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3237), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3264), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3303),  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3429),  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3479), and  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3873).

**6.131.3.5 virtual void** `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal`  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1652).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`  
 (p. 224), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
 (p. 272), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
 (p. 393), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
 (p. 420), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
 (p. 465), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
 (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`

(p. 690), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 927), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1295), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1303), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1364), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1409), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1471), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1500), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1534), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1597), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1735), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1849), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1947), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2101), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2276), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2572), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2609), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2642), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2747), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3043), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3108), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3201), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3238), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3265), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3304), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3430), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3480), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3874).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

## 6.132 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference

Base class for all Marshallers that **marshal** (p. 107) DataStructures to and from the wire using the OpenWire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits **activemq::wireformat::openwire::marshal::DataStreamMarshaller**.

Inherited by **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**,

**activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**,

**activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**,

**activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**,

**activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**,

**activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**,

**activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**,

**activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**,

**activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**,

**activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**,

**activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**,

**activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**,

**activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**, ac-

**tivemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**, ac-

tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller,	ac-

```

tivemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller,          ac-
tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller,
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,           ac-
tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller,
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,          ac-
tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,             ac-
tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,      ac-
tivemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,         and ac-
tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller.

```

## Public Member Functions

- virtual `~BaseDataStreamMarshaller` ()
- virtual `int tightMarshal1 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Marshal to the given stream.*
- virtual `void tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Marshal to the given stream.*
- virtual `void tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Un-Marshal to the given stream.*
- virtual `void looseMarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Marshal to the given stream.*
- virtual `void looseUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Loose Un-Marshal to the given stream.*

## Static Public Member Functions

- static `std::string toString (const commands::MessageId *id)`  
*Converts the object to a String.*
- static `std::string toString (const commands::ProducerId *id)`  
*Converts the object to a String.*



- static `std::string toString` (`const commands::TransactionId *txnId`)  
*Converts the given transaction ID into a String.*
- static `std::string toHexFromBytes` (`const std::vector< unsigned char > &data`)  
*given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.*

## Protected Member Functions

- virtual `commands::DataStructure * tightUnmarshalCachedObject` (`OpenWireFormat *wireFormat`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tight Unmarshal the cached object.*
- virtual `int tightMarshalCachedObject1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *data`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*
- virtual `void tightMarshalCachedObject2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *data`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*
- virtual `void looseMarshalCachedObject` (`OpenWireFormat *wireFormat`, `commands::DataStructure *data`, `decaf::io::DataOutputStream *dataOut`) throw (`decaf::io::IOException`)  
*Loosely marshals the passed DataStructure based object to the passed stream returning nothing.*
- virtual `commands::DataStructure * looseUnmarshalCachedObject` (`OpenWireFormat *wireFormat`, `decaf::io::DataInputStream *dataIn`) throw (`decaf::io::IOException`)  
*Loose Unmarshal the cached object.*
- virtual `int tightMarshalNestedObject1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *object`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*
- virtual `void tightMarshalNestedObject2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *object`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*
- virtual `commands::DataStructure * tightUnmarshalNestedObject` (`OpenWireFormat *wireFormat`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)  
*Tight Unmarshal the nested object.*

- virtual **commands::DataStructure \* looseUnmarshalNestedObject** (**OpenWireFormat \*wireFormat**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )

*Loose Unmarshal the nested object.*

- virtual void **looseMarshalNestedObject** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*object**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Loose marshal the nested object.*

- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Performs Tight Unmarshaling of String Objects.*

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.*

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Tight Marshals the passed string to the streams passed.*

- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Loose Marshal the String to the DataOuputStream passed.*

- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )

*Loose Un-Marshal the String to the DataOuputStream passed.*

- virtual int **tightMarshalLong1** (**OpenWireFormat \*wireFormat**, long long value, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Tightly marshal (p. 107) the long long to the BooleanStream passed.*

- virtual void **tightMarshalLong2** (**OpenWireFormat \*wireFormat**, long long value, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Tightly marshal (p. 107) the long long to the Streams passed.*

- virtual long long **tightUnmarshalLong** (**OpenWireFormat \*wireFormat**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Tight marshal (p. 107) the long long type.*

- virtual void **looseMarshalLong** (**OpenWireFormat \*wireFormat**, long long value, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Tightly marshal (p. 107) the long long to the BooleanStream passed.*

- virtual long long **looseUnmarshalLong** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose marshal (p. 107) the long long type.*
- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Unmarshal an array of char.*
- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose Unmarshal an array of char.*
- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs, int size) throw ( **decaf::io::IOException** )  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*
- virtual std::vector< unsigned char > **looseUnmarshalConstByteArray** (**decaf::io::DataInputStream** \*dataIn, int size) throw ( **decaf::io::IOException** )  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*
- virtual **commands::DataStructure** \* **tightUnmarshalBrokerError** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Unmarshal the Error object.*
- virtual int **tightMarshalBrokerError1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*
- virtual void **tightMarshalBrokerError2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*
- virtual **commands::DataStructure** \* **looseUnmarshalBrokerError** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose Unmarshal the Error object.*
- virtual void **looseMarshalBrokerError** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*
- template<typename T >  
int **tightMarshalObjectArray1** (**OpenWireFormat** \*wireFormat, std::vector< T > objects, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.*

- `template<typename T >`  
`void tightMarshalObjectArray2 (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`

*Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*

- `template<typename T >`  
`void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`

*Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*

- `virtual std::string readAsciiString (decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`

*Given an DataInputStream read a know ASCII formatted string from the input and return that string.*

### 6.132.1 Detailed Description

Base class for all Marshallers that **marshal** (p.107) DataStructures to and from the wire using the OpenWire protocol.

Since:

2.0

### 6.132.2 Constructor & Destructor Documentation

- 6.132.2.1 `virtual`  
`activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller () [inline, virtual]`

### 6.132.3 Member Function Documentation

- 6.132.3.1 `virtual void activate (activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED) throw ( decaf::io::IOException ) [inline, virtual]`

Tight Marshal to the given stream.

Parameters:

*format* - The OpenwireFormat properties

*command* - the object to Marshal

*ds* - DataOutputStream to **marshal** (p.107) to

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.2** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerE  
(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*,  
decaf::io::DataOutputStream \* *dataOut*) throw ( decaf::io::IOException )  
[protected, virtual]

Tight Marshal the Error object.

#### Parameters:

*wireFormat* - The OpenWireFormat properties

*data* - Error to Marshal

*dataOut* - stream to write marshalled data to

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.3** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedO  
(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*,  
decaf::io::DataOutputStream \* *dataOut*) throw ( decaf::io::IOException )  
[protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

#### Parameters:

*wireFormat* - The OpenWireFormat properties

*data* - DataStructure Object Pointer to **marshal** (p.107)

*dataOut* - stream to write marshaled data to

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.4** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong  
(OpenWireFormat \* *wireFormat*, long long *value*, de-  
caf::io::DataOutputStream \* *dataOut*) throw ( decaf::io::IOException )  
[protected, virtual]

Tightly **marshal** (p.107) the long long to the BooleanStream passed.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*value* - long long to **marshal** (p. 107)

*dataOut* - DataOutputStream to **marshal** (p. 107) to.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.5** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject(OpenWireFormat * wireFormat, commands::DataStructure * object, decaf::io::DataOutputStream * dataOut) throw ( decaf::io::IOException )`  
[protected, virtual]

Loose marshal the nested object.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*object* - DataStructure Object Pointer to **marshal** (p. 107)

*dataOut* - stream to write marshaled data to

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.6** `template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray(OpenWireFormat * wireFormat, std::vector< T > objects, decaf::io::DataOutputStream * dataOut) throw ( decaf::io::IOException )`  
[inline, protected]

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*objects* - array of DataStructure object pointers.

*dataOut* - stream to write marshalled data to

**Returns:**

size of the marshalled data

**Exceptions:**

*IOException* if an error occurs.

References `AMQ_CATCH_EXCEPTION_CONVERT`, `AMQ_CATCH_RETHROW`, and `AMQ_CATCHALL_THROW`.

**6.132.3.7** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString (const std::string *value*, decaf::io::DataOutputStream \* *dataOut*) throw ( decaf::io::IOException ) [protected, virtual]

Loose Marshal the String to the DataOutputStream passed.

**Parameters:**

*value* - string to **marshal** (p. 107)

*dataOut* - stream to write marshaled form to

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.8** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat \**format* *AMQCPP\_UNUSED*, commands::DataStructure \**command* *AMQCPP\_UNUSED*, decaf::io::DataInputStream \**dis* *AMQCPP\_UNUSED*) throw ( decaf::io::IOException ) [inline, virtual]

Loose Un-Marshal to the given stream.

**Parameters:**

*format* - The OpenWireFormat properties

*command* - the object to Un-Marshal

*dis* - the DataInputStream to Un-Marshal from

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.9** virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBroken (OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [protected, virtual]

Loose Unmarshal the Error object.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshalled form from

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.10** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBytes (decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [protected, virtual]

Loose Unmarshal an array of char.

**Parameters:**

*dataIn* - the DataInputStream to Un-Marshall from

**Returns:**

the unmarshalled vector of chars.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.11** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [protected, virtual]

Loose Unmarshal the cached object.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.12** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCompact (decaf::io::DataInputStream * dataIn, int size) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

**Parameters:**

*dataIn* - the DataInputStream to Un-Marshall from

*size* - size of the const array to unmarshal

**Returns:**

the unmarshaled vector of chars.



**Exceptions:**

*IOException* if an error occurs.

**6.132.3.13** virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [protected, virtual]

Loose marshal (p.107) the long long type.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

**Returns:**

the unmarshaled long long

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.14** virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNested (OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [protected, virtual]

Loose Unmarshal the nested object.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.15** virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

**Parameters:**

*dataIn* - stream to read marshaled form from

**Returns:**

the unmarshaled string

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.16** `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )`  
[protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

**Parameters:**

*dataIn* - DataInputStream to read from

**Returns:**

string value read from stream

**6.132.3.17** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )` [inline, virtual]

Tight Marshal to the given stream.

**Parameters:**

*format* - The OpenWireFormat properties

*command* - the object to Marshal

*bs* - boolean stream to **marshal** (p.107) to.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.18** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )` [inline, virtual]

Tight Marshal to the given stream.

**Parameters:**

*format* - The OpenwireFormat properties  
*command* - the object to Marshal  
*ds* - the DataOutputStream to Marshal to  
*bs* - boolean stream to **marshal** (p.107) to.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.19** virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tight Marshal the Error object.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*data* - Error to Marshal  
*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

size of the marshalled data

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.20** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tight Marshal the Error object.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*data* - Error to Marshal  
*dataOut* - stream to write marshalled data to  
*bs* - boolean stream to **marshal** (p.107) to.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.21** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached (OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*data* - DataStructure Object Pointer to **marshal** (p.107)

*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

size of data written.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.22** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached (OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*data* - DataStructure Object Pointer to **marshal** (p.107)

*bs* - boolean stream to **marshal** (p.107) to.

*dataOut* - stream to write marshaled data to

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.23** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1 (OpenWireFormat * wireFormat, long long value, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tightly **marshal** (p.107) the long long to the BooleanStream passed.

**Parameters:**

*wireFormat* - The OpenwireFormat properties

*value* - long long to **marshal** (p. 107)

*bs* - boolean stream to **marshal** (p. 107) to.

#### Returns:

size of data written.

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.24** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2 (OpenWireFormat \* *wireFormat*, long long *value*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tightly **marshal** (p. 107) the long long to the Streams passed.

#### Parameters:

*wireFormat* - The OpenWireFormat properties

*value* - long long to **marshal** (p. 107)

*dataOut* - stream to write marshaled form to

*bs* - boolean stream to **marshal** (p. 107) to.

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.25** virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *object*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

#### Parameters:

*wireFormat* - The OpenWireFormat properties

*object* - DataStructure Object Pointer to **marshal** (p. 107)

*bs* - boolean stream to **marshal** (p. 107) to.

#### Returns:

size of data written.

#### Exceptions:

*IOException* if an error occurs.

**6.132.3.26** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested(OpenWireFormat * wireFormat, commands::DataStructure * object, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*object* - DataStructure Object Pointer to **marshal** (p.107)  
*bs* - boolean stream to **marshal** (p.107) to.  
*dataOut* - stream to write marshaled data to

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.27** `template<typename T > int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject(OpenWireFormat * wireFormat, std::vector< T > objects, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [inline, protected]

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*objects* - array of DataStructure object pointers.  
*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

size of the marshalled data

**Exceptions:**

*IOException* if an error occurs.

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

**6.132.3.28** `template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject(OpenWireFormat * wireFormat, std::vector< T > objects, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [inline, protected]

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*objects* - array of DataStructure object pointers.  
*dataOut* - stream to write marshalled data to  
*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

size of the marshalled data

**Exceptions:**

*IOException* if an error occurs.

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

**6.132.3.29** virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & *value*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

**Parameters:**

*value* - string to **marshal** (p.107)  
*bs* - BooleanStream to use.

**Returns:**

size of marshaled string.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.30** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & *value*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [protected, virtual]

Tight Marshals the passed string to the streams passed.

**Parameters:**

*value* - string to **marshal** (p.107)  
*dataOut* - the DataOutputStream to Marshal to  
*bs* - boolean stream to **marshal** (p.107) to.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.31** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )` [inline, virtual]

Tight Un-Marshall to the given stream.

**Parameters:**

*format* - The OpenwireFormat properties  
*command* - the object to Un-Marshall  
*dis* - the DataInputStream to Un-Marshall from  
*bs* - boolean stream to Un-Marshall from.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.32** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBroken (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshall the Error object.

**Parameters:**

*wireFormat* - The OpenwireFormat properties  
*dataIn* - stream to read marshalled form from  
*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.33** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBytes (decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal an array of char.

**Parameters:**

*dataIn* - the DataInputStream to Un-Marshall from



*bs* - boolean stream to unmarshal from.

**Returns:**

the unmarshaled vector of chars.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.34** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCached(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal the cached object.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.35** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConstArray(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

**Parameters:**

*dataIn* - the DataInputStream to Un-Marshall from

*bs* - boolean stream to unmarshal from.

*size* - size of the const array to unmarshal

**Returns:**

the unmarshaled vector of chars.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.36** `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tight **marshal** (p.107) the long long type.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

the unmarshaled long long

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.37** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNested (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal the nested object.

**Parameters:**

*wireFormat* - The OpenWireFormat properties

*dataIn* - stream to read marshaled form from

*bs* - boolean stream to **marshal** (p.107) to.

**Returns:**

pointer to a new DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.38** `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString (decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [protected, virtual]

Performs Tight Unmarshaling of String Objects.

**Parameters:**

*dataIn* - the DataInputStream to Un-Marshall from

*bs* - boolean stream to unmarshal from.

**Returns:**

the unmarshaled string.

**Exceptions:**

*IOException* if an error occurs.

**6.132.3.39** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes(const std::vector< unsigned char > & data) [static]`

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

**Parameters:**

*data* - unsigned char data array pointer

**Returns:**

a string coded in hex that represents the data

**6.132.3.40** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString(const commands::TransactionId * txnId) [static]`

Converts the given transaction ID into a String.

**Parameters:**

*txnId* - TransactionId poitner

**Returns:**

string representation of the id

**6.132.3.41** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString(const commands::ProducerId * id) [static]`

Converts the object to a String.

**Parameters:**

*id* - ProducerId pointer

**Returns:**

string representing the id

**6.132.3.42** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString(const commands::MessageId * id) [static]`

Converts the object to a String.

**Parameters:**

*id* - MessageId pointer

**Returns:**

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

## 6.133 activemq::commands::BaseDataStructure Class Reference

#include <src/main/activemq/commands/BaseDataStructure.h> Inheritance diagram for activemq::commands::BaseDataStructure:

### Public Member Functions

- virtual `~BaseDataStructure ()`
- virtual bool `isMarshalAware ()` const  
*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*
- virtual void `beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw ( decaf::io::IOException )  
*Perform any processing needed before an marshal.*
- virtual void `afterMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*
- virtual void `beforeUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw ( decaf::io::IOException )  
*Perform any processing needed before an unmarshal.*
- virtual void `afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*
- virtual void `setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)`  
*Called to set the data to this object that will contain the objects marshaled form.*
- virtual std::vector< unsigned char > `getMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)`  
*Called to get the data to this object that will contain the objects marshaled form.*
- virtual void `copyDataStructure (const DataStructure *src AMQCPP_UNUSED)`  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string `toString ()` const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value AMQCPP_UNUSED)` const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*

### 6.133.1 Constructor & Destructor Documentation

**6.133.1.1** `virtual activemq::commands::BaseDataStructure::~~BaseDataStructure ()`  
`[inline, virtual]`

### 6.133.2 Member Function Documentation

**6.133.2.1** `virtual void activemq::commands::BaseDataStructure::afterMarshal`  
`(wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (`  
`decaf::io::IOException ) [inline, virtual]`

Perform any processing needed after an unmarshal.

**Parameters:**

*wireformat* (p. 105) - the OpenWireFormat object in use.

**6.133.2.2** `virtual void activemq::commands::BaseDataStructure::afterUnmarshal`  
`(wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (`  
`decaf::io::IOException ) [inline, virtual]`

Perform any processing needed after an unmarshal.

**Parameters:**

*wireformat* (p. 105) - the OpenWireFormat object in use.

Reimplemented in `activemq::commands::Message` (p. 2520), and `activemq::commands::WireFormatInfo` (p. 3984).

**6.133.2.3** `virtual void activemq::commands::BaseDataStructure::beforeMarshal`  
`(wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (`  
`decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an marshal.

**Parameters:**

*wireformat* (p. 105) - the OpenWireFormat object in use.

Reimplemented in `activemq::commands::Message` (p. 2520), and `activemq::commands::WireFormatInfo` (p. 3985).

**6.133.2.4** `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal`  
`(wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (`  
`decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an unmarshal.

**Parameters:**

*wireformat* (p. 105) - the OpenWireFormat object in use.

### 6.133.2.5 virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure \*src *AMQCPP\_UNUSED*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns:

src - Source Object

Reimplemented in `activemq::commands::BooleanExpression` (p. 856).

### 6.133.2.6 virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure \*value *AMQCPP\_UNUSED*) const [inline, virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

### 6.133.2.7 virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat \*wireFormat *AMQCPP\_UNUSED*) [inline, virtual]

Called to get the data to this object that will contain the objects marshaled form.

#### Parameters:

*wireFormat* - the `wireformat` (p. 105) object to control unmarshaling

#### Returns:

buffer that holds the objects data.

### 6.133.2.8 virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

#### Returns:

true if aware of marshaling

Implements `activemq::wireformat::MarshalAware` (p. 2486).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 368), `activemq::commands::Message` (p. 2526), and `activemq::commands::WireFormatInfo` (p. 3987).

**6.133.2.9** `virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)` [inline, virtual]

Called to set the data to this object that will contain the objects marshaled form.

**Parameters:**

*wireFormat* - the **wireformat** (p.105) object to control unmarshaling

*data* - vector of object binary data

**6.133.2.10** `virtual std::string activemq::commands::BaseDataStructure::toString () const` [inline, virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p.1663).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p.208), **activemq::commands::ActiveMQBytesMessage** (p.244), **activemq::commands::ActiveMQDestination** (p.330), **activemq::commands::ActiveMQMapMessage** (p.372), **activemq::commands::ActiveMQMessage** (p.400), **activemq::commands::ActiveMQObjectMessage** (p.445), **activemq::commands::ActiveMQQueue** (p.486), **activemq::commands::ActiveMQStreamMessage** (p.547), **activemq::commands::ActiveMQTempDestination** (p.578), **activemq::commands::ActiveMQTempQueue** (p.607), **activemq::commands::ActiveMQTempTopic** (p.636), **activemq::commands::ActiveMQTextMessage** (p.666), **activemq::commands::ActiveMQTopic** (p.694), **activemq::commands::BaseCommand** (p.763), **activemq::commands::BooleanExpression** (p.856), **activemq::commands::BrokerId** (p.871), **activemq::commands::BrokerInfo** (p.901), **activemq::commands::Command** (p.1198), **activemq::commands::ConnectionControl** (p.1270), **activemq::commands::ConnectionError** (p.1298), **activemq::commands::ConnectionId** (p.1329), **activemq::commands::ConnectionInfo** (p.1359), **activemq::commands::ConsumerControl** (p.1404), **activemq::commands::ConsumerId** (p.1433), **activemq::commands::ConsumerInfo** (p.1464), **activemq::commands::ControlCommand** (p.1495), **activemq::commands::DataArrayResponse** (p.1530), **activemq::commands::DataResponse** (p.1585), **activemq::commands::DestinationInfo** (p.1730), **activemq::commands::DiscoveryEvent** (p.1760), **activemq::commands::ExceptionResponse** (p.1841), **activemq::commands::FlushCommand** (p.1939), **activemq::commands::IntegerResponse** (p.2093), **activemq::commands::JournalQueueAck** (p.2158), **activemq::commands::JournalTopicAck** (p.2186), **activemq::commands::JournalTrace** (p.2214), **activemq::commands::JournalTransaction** (p.2241), **activemq::commands::KeepAliveInfo** (p.2268), **activemq::commands::LastPartialCommand** (p.2302), **activemq::commands::LocalTransactionId** (p.2350), **activemq::commands::Message** (p.2530), **activemq::commands::MessageAck**



(p. 2563), [activemq::commands::MessageDispatch](#) (p. 2597), [activemq::commands::MessageDispatchNotification](#) (p. 2633), [activemq::commands::MessageId](#) (p. 2666), [activemq::commands::MessagePull](#) (p. 2742), [activemq::commands::NetworkBridgeFilter](#) (p. 2795), [activemq::commands::PartialCommand](#) (p. 2920), [activemq::commands::ProducerAck](#) (p. 3038), [activemq::commands::ProducerId](#) (p. 3070), [activemq::commands::ProducerInfo](#) (p. 3099), [activemq::commands::RemoveInfo](#) (p. 3196), [activemq::commands::RemoveSubscriptionInfo](#) (p. 3225), [activemq::commands::ReplayCommand](#) (p. 3253), [activemq::commands::Response](#) (p. 3287), [activemq::commands::SessionId](#) (p. 3382), [activemq::commands::SessionInfo](#) (p. 3409), [activemq::commands::ShutdownInfo](#) (p. 3472), [activemq::commands::SubscriptionInfo](#) (p. 3674), [activemq::commands::TransactionId](#) (p. 3821), [activemq::commands::TransactionInfo](#) (p. 3849), [activemq::commands::WireFormatInfo](#) (p. 3991), and [activemq::commands::XATransactionId](#) (p. 4034).

Referenced by [activemq::commands::BooleanExpression::toString\(\)](#), and [activemq::commands::BaseCommand::toString\(\)](#).

The documentation for this class was generated from the following file:

- [src/main/activemq/commands/BaseDataStructure.h](#)

## 6.134 `binary_function` Class Reference

Inheritance diagram for `binary_function`:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

## 6.135 decaf::net::BindException Class Reference

#include <src/main/decaf/net/BindException.h> Inheritance diagram for decaf::net::BindException:

### Public Member Functions

- **BindException** () throw ()  
*Default Constructor.*
- **BindException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BindException** (const **BindException** &ex) throw ()  
*Copy Constructor.*
- **BindException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BindException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BindException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BindException** \* clone () const  
*Clones this exception.*
- virtual ~**BindException** () throw ()

### 6.135.1 Constructor & Destructor Documentation

#### 6.135.1.1 decaf::net::BindException::BindException () throw () [inline]

Default Constructor.

#### 6.135.1.2 decaf::net::BindException::BindException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

**6.135.1.3** `decaf::net::BindException::BindException (const BindException & ex)  
throw () [inline]`

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.135.1.4** `decaf::net::BindException::BindException (const char * file, const int  
lineNumber, const std::exception * cause, const char * msg, ...) throw  
() [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.135.1.5** `decaf::net::BindException::BindException (const std::exception * cause)  
throw () [inline]`

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.135.1.6** `decaf::net::BindException::BindException (const char * file, const int  
lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.135.1.7** `virtual decaf::net::BindException::~~BindException () throw () [inline, virtual]`

## 6.135.2 Member Function Documentation

**6.135.2.1** `virtual BindException* decaf::net::BindException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3522).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

## 6.136 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

#include <src/main/decaf/io/BlockingByteArrayInputStream.h> Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

### Public Member Functions

- **BlockingByteArrayInputStream** ()  
*Default Constructor - uses a default **internal** (p. 144) buffer.*
- **BlockingByteArrayInputStream** (const unsigned char \*buffer, int bufferSize)  
*Constructor that initializes the **internal** (p. 144) buffer.*
- virtual ~**BlockingByteArrayInputStream** ()
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize)
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.*  
**Returns:**  
*the number of bytes available on this input stream.*  
**Exceptions:**  
***IOException** (p. 2142) if an I/O error occurs.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2043) freeing any resources that might have been acquired during the lifetime of this stream.  
The default implementation of this method does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.  
The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*  
**Parameters:**  
*num The number of bytes to skip.*

**Returns:***total bytes skipped***Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*  
***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( **IOException** )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.136.1 Detailed Description**

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the **internal** (p. 144) buffer via a call to **setByteArray**.

**6.136.2 Constructor & Destructor Documentation****6.136.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()**

Default Constructor - uses a default **internal** (p. 144) buffer.

**6.136.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char \* *buffer*, int *bufferSize*)**

Constructor that initializes the **internal** (p. 144) buffer.

See also:

**setByteArray** (p. 841).

**6.136.2.3 virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream () [virtual]****6.136.3 Member Function Documentation****6.136.3.1 virtual int decaf::io::BlockingByteArrayInputStream::available () const throw ( decaf::io::IOException ) [virtual]**

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply

return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns:**

the number of bytes available on this input stream.

**Exceptions:**

***IOException*** (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.136.3.2** `virtual void decaf::io::BlockingByteArrayInputStream::close () throw ( decaf::io::IOException )` [virtual]

Closes the **InputStream** (p. 2043) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.136.3.3** `virtual int decaf::io::BlockingByteArrayInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2046).

**6.136.3.4** `virtual int decaf::io::BlockingByteArrayInputStream::doReadByte () throw ( IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2046).

**6.136.3.5** `virtual void decaf::io::BlockingByteArrayInputStream::setByteArray (const unsigned char * buffer, int bufferSize)` [virtual]

**6.136.3.6** `virtual long long decaf::io::BlockingByteArrayInputStream::skip (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.



**Parameters:**

*num* The number of bytes to skip.

**Returns:**

total bytes skipped

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

*UnsupportedOperationException* if the concrete stream class does not support skipping bytes.

Reimplemented from `decaf::io::InputStream` (p. 2050).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BlockingByteArrayInputStream.h`

## 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference

A **decaf::util::Queue** (p. 3149) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

#include <src/main/decaf/util/concurrent/BlockingQueue.h> Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

### Public Member Functions

- virtual **~BlockingQueue** ()
- virtual void **put** (const E &value)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting if necessary for space to become available.*
- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.*
- virtual E **take** ()=0 throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.*
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.*
- virtual int **remainingCapacity** () const =0  
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX\_VALUE** if there is no intrinsic limit.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes all available elements from this queue and adds them to the given collection.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes at most the given number of available elements from this queue and adds them to the given collection.*

### 6.137.1 Detailed Description

`template<typename E> class decaf::util::concurrent::BlockingQueue< E >`

A **decaf::util::Queue** (p. 3149) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element. **BlockingQueue** (p. 843) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either **true** or **false**, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

	<i>Throws exception</i>	<i>Boolean Flag</i>	<i>Blocks</i>	<i>Times out</i>
<b>Insert</b>	<b>add(e)</b> (p. 195)	<b>offer(e)</b> (p. 847)	<b>put(e)</b> (p. 848)	<b>offer(e, time, unit)</b> (p. ??)
<b>Remove</b>	<b>remove()</b> (p. 196)	<b>poll()</b> (p. 847)	<b>take()</b> (p. 849)	<b>poll(time, unit)</b> (p. ??)
<b>Examine</b>	<b>element()</b> (p. 196)	<b>peek()</b> (p. 3151)	<i>not applicable</i>	<i>not applicable</i>

A **BlockingQueue** (p. 843) may be capacity bounded. At any given time it may have a **remainingCapacity** beyond which no additional elements can be **put** without blocking. A **BlockingQueue** (p. 843) without any intrinsic capacity constraints always reports a remaining capacity of **Integer::MAX\_VALUE**.

**BlockingQueue** (p. 843) implementations are designed to be used primarily for producer-consumer queues, but additionally support **decaf::util::Collection** (p. 1184) interface. So, for example, it is possible to remove an arbitrary element from a queue using **remove(x)**. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

**BlockingQueue** (p. 843) implementations are thread-safe. All queuing methods achieve their effects atomically using **internal** (p. 144) **locks** (p. 174) or other forms of concurrency control. However, the *bulk* **Collection** (p. 1184) operations **addAll**, **containsAll**, **retainAll** and **removeAll** are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for **addAll(c)** to fail (throwing an exception) after adding only some of the elements in **c**.

A **BlockingQueue** (p. 843) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a **BlockingQueue** (p. 843) can safely be used with multiple producers and multiple consumers.

```
class Producer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}
```

```

    virtual void run() {
        try {
            while( true ) { queue->put( produce() ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    Object produce() { ... }
}

class Consumer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { consume( queue->take() (p.849) ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    void consume( Object& x ) { ... }
}

int main( int argc, char** argv ) {

    BlockingQueue (p.843) q = new SomeQueueImplementation();
    Producer p( &q );
    Consumer c1( &q );
    Consumer c2( &q );
    Thread t1( &p ).start();
    Thread t2( &c1 ).start();
    Thread t3( &c2 ).start();
}

```

Memory consistency effects: As with other **concurrent** (p.171) collections, actions in a thread prior to placing an object into a **BlockingQueue** (p.843) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p.843) in another thread.

**Since:**

1.0

## 6.137.2 Constructor & Destructor Documentation

**6.137.2.1** `template<typename E > virtual decaf::util::concurrent::BlockingQueue< E >::~~BlockingQueue () [inline, virtual]`

## 6.137.3 Member Function Documentation

**6.137.3.1** `template<typename E > virtual std::size_t decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & c, std::size_t maxElements) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes at most the given number of available elements from this queue and adds them to the given collection. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

### Parameters:

*c* the collection to transfer elements into  
*maxElements* the maximum number of elements to transfer

### Returns:

the number of elements transferred

### Exceptions:

***UnsupportedOperationException*** if addition of elements is not supported by the specified collection  
***IllegalArgumentException*** if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3719).

**6.137.3.2** `template<typename E > virtual std::size_t decaf::util::concurrent::BlockingQueue< E >::drainTo (Collection< E > & c) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes all available elements from this queue and adds them to the given collection. This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

### Parameters:

*c* the collection to transfer elements into

### Returns:

the number of elements transferred

**Exceptions:**

***UnsupportedOperationException*** if addition of elements is not supported by the specified collection

***IllegalArgumentExcepion*** if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3720).

**6.137.3.3** `template<typename E > virtual bool  
decaf::util::concurrent::BlockingQueue< E >::offer  
(const E & e, long timeout, const TimeUnit & unit)  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException, de-  
cafe::lang::exceptions::IllegalArgumentExcepion ) [pure  
virtual]`

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

**Parameters:**

*e* the element to add

*timeout* how long to wait before giving up, in units of *unit*

*unit* a `TimeUnit` (p. 3807) determining how to interpret the *timeout* parameter

**Returns:**

`true` if successful, or `false` if the specified waiting time elapses before space is available

**Exceptions:**

***InterruptedException*** if interrupted while waiting

***NullPointerException*** if the specified element is null

***IllegalArgumentExcepion*** if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3722).

**6.137.3.4** `template<typename E > virtual bool  
decaf::util::concurrent::BlockingQueue< E >::poll (E &  
result, long long timeout, const TimeUnit & unit) throw (  
decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

**Parameters:**

*result* the referenced value that will be assigned the value retrieved from the **Queue** (p. 3149). Undefined if this methods returned false.

*timeout* how long to wait before giving up, in units of *unit*

*unit* a TimeUnit (p. 3807) determining how to interpret the `timeout` parameter.

**Returns:**

`true` if successful or `false` if the specified waiting time elapses before an element is available.

**Exceptions:**

*InterruptedException* if interrupted while waiting

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3722).

**6.137.3.5** `template<typename E > virtual void  
decaf::util::concurrent::BlockingQueue< E >::put (const E  
& value) throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException, de-  
cafe::lang::exceptions::IllegalArgumentException ) [pure  
virtual]`

Inserts the specified element into this queue, waiting if necessary for space to become available.

**Parameters:**

*value* the element to add

**Exceptions:**

*InterruptedException* if interrupted while waiting

*NullPointerException* if the specified element is null

*IllegalArgumentException* if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3723).

**6.137.3.6** `template<typename E > virtual int  
decaf::util::concurrent::BlockingQueue< E >::remainingCapacity () const  
[pure virtual]`

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit. Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

**Returns:**

the remaining capacity

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3723).

```
6.137.3.7  template<typename E > virtual E  
            decaf::util::concurrent::BlockingQueue< E >::take ()  
            throw ( decaf::lang::exceptions::InterruptedException ) [pure virtual]
```

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

**Returns:**

the head of this queue

**Exceptions:**

*InterruptedException* if interrupted while waiting

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3724).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BlockingQueue.h**



## 6.138 decaf::lang::Boolean Class Reference

#include <src/main/decaf/lang/Boolean.h> Inheritance diagram for decaf::lang::Boolean:

### Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual ~**Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const  
*Compares this **Boolean** (p. 850) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Boolean** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const  
*Compares this **Boolean** (p. 850) instance with another.*
- virtual bool **operator==** (const bool &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const bool &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const bool &b) const

### Static Public Member Functions

- static **Boolean** **valueOf** (bool value)
- static **Boolean** **valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)  
*Parses the **String** (p. 3665) passed and extracts an bool.*
- static std::string **toString** (bool value)  
*Converts the bool to a **String** (p. 3665) representation.*

## Static Public Attributes

- static const **Boolean** **\_FALSE**

*The Class object representing the primitive false boolean.*

- static const **Boolean** **\_TRUE**

*The Class object representing the primitive type boolean.*

## 6.138.1 Constructor & Destructor Documentation

### 6.138.1.1 `decaf::lang::Boolean::Boolean (bool value)`

#### Parameters:

*value* - primitive boolean to wrap.

### 6.138.1.2 `decaf::lang::Boolean::Boolean (const std::string & value)`

#### Parameters:

*value* - **String** (p. 3665) value to convert to a boolean.

### 6.138.1.3 `virtual decaf::lang::Boolean::~~Boolean ()` [inline, virtual]

## 6.138.2 Member Function Documentation

### 6.138.2.1 `bool decaf::lang::Boolean::booleanValue () const` [inline]

#### Returns:

the primitive boolean value of this object

### 6.138.2.2 `virtual int decaf::lang::Boolean::compareTo (const bool & b) const` [virtual]

Compares this **Boolean** (p. 850) instance with another.

#### Parameters:

*b* - the **Boolean** (p. 850) instance to be compared

#### Returns:

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements `decaf::lang::Comparable< bool >` (p. 1214).

**6.138.2.3** `virtual int decaf::lang::Boolean::compareTo (const Boolean & b) const`  
[virtual]

Compares this **Boolean** (p. 850) instance with another.

**Parameters:**

*b* - the **Boolean** (p. 850) instance to be compared

**Returns:**

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

**6.138.2.4** `bool decaf::lang::Boolean::equals (const bool & b) const` [inline,  
virtual]**Returns:**

true if the two **Boolean** (p. 850) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1215).

**6.138.2.5** `bool decaf::lang::Boolean::equals (const Boolean & b) const` [inline]**Returns:**

true if the two **Boolean** (p. 850) Objects have the same value.

**6.138.2.6** `virtual bool decaf::lang::Boolean::operator< (const bool & value) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1215).

**6.138.2.7** `virtual bool decaf::lang::Boolean::operator< (const Boolean & value)`  
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.138.2.8** `virtual bool decaf::lang::Boolean::operator==(const bool & value) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< bool >` (p. 1215).

**6.138.2.9** `virtual bool decaf::lang::Boolean::operator==(const Boolean & value)`  
`const` [virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.138.2.10** `static bool decaf::lang::Boolean::parseBoolean(const std::string &`  
`value)` [static]

Parses the **String** (p. 3665) passed and extracts an bool.

**Parameters:**

*value* The std::string value to parse

**Returns:**

bool value

**6.138.2.11 static std::string decaf::lang::Boolean::toString (bool *value*) [static]**

Converts the bool to a **String** (p. 3665) representation.

**Parameters:**

*value* The bool value to convert.

**Returns:**

std::string representation of the bool value passed.

**6.138.2.12 std::string decaf::lang::Boolean::toString () const****Returns:**

the string representation of this Booleans value.

**6.138.2.13 static Boolean decaf::lang::Boolean::valueOf (const std::string & *value*) [static]****Parameters:**

*value* The std::string value to convert to a Boolean (p. 850) instance.

**Returns:**

a **Boolean** (p. 850) instance of the string value

**6.138.2.14 static Boolean decaf::lang::Boolean::valueOf (bool *value*) [static]****Parameters:**

*value* The bool value to convert to a Boolean (p. 850) instance.

**Returns:**

a **Boolean** (p. 850) instance of the primitive boolean value

**6.138.3 Field Documentation****6.138.3.1 const Boolean decaf::lang::Boolean::\_FALSE [static]**

The Class object representing the primitive false boolean.

**6.138.3.2 const Boolean decaf::lang::Boolean::\_TRUE [static]**

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

## 6.139 activemq::commands::BooleanExpression Class Reference

#include <src/main/activemq/commands/BooleanExpression.h> Inheritance diagram for activemq::commands::BooleanExpression:

### Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStream \* cloneDataStream** () const  
*Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStream** (const **DataStream** \*src AMQCPP\_UNUSED)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStream** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStream** \*value) const  
*Compares the **DataStream** (p. 1660) passed in to this one, and returns if they are equivalent.*

### 6.139.1 Constructor & Destructor Documentation

- 6.139.1.1** **activemq::commands::BooleanExpression::BooleanExpression** ()  
[inline]
- 6.139.1.2** **virtual activemq::commands::BooleanExpression::~~BooleanExpression** ()  
[inline, virtual]

### 6.139.2 Member Function Documentation

- 6.139.2.1** **virtual DataStream\* activemq::commands::BooleanExpression::cloneDataStream** () const  
[inline, virtual]

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements **activemq::commands::DataStream** (p. 1660).

**6.139.2.2 virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure \*src *AMQCPP\_UNUSED*) [inline, virtual]**

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from **activemq::commands::BaseDataStructure** (p. 832).

**6.139.2.3 virtual bool activemq::commands::BooleanExpression::equals (const DataStructure \* *value*) const [inline, virtual]**

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

References **activemq::commands::BaseDataStructure::equals()**.

**6.139.2.4 virtual std::string activemq::commands::BooleanExpression::toString () const [inline, virtual]**

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 833).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

## 6.140 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** ()
- bool **readBoolean** () throw ( decaf::io::IOException )  
*Read a boolean data element from the internal data buffer.*
- void **writeBoolean** (bool value) throw ( decaf::io::IOException )  
*Writes a Boolean value to the internal data buffer.*
- void **marshal** (decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )  
*Marshal the data to a DataOutputStream.*
- void **marshal** (std::vector< unsigned char > &dataOut)  
*Marshal the data to a STL vector of unsigned chars.*
- void **unmarshal** (decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )  
*Unmarshal a Boolean data stream from the Input Stream.*
- void **clear** ()  
*Clears to old position markers, data starts at the beginning.*
- int **marshalledSize** ()  
*Calc the size that data is marshalled to.*

### 6.140.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.



## 6.140.2 Constructor & Destructor Documentation

**6.140.2.1** `activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ()`

**6.140.2.2** `virtual  
activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream  
( ) [inline, virtual]`

## 6.140.3 Member Function Documentation

**6.140.3.1** `void activemq::wireformat::openwire::utils::BooleanStream::clear ()`

Clears to old position markers, data starts at the beginning.

**6.140.3.2** `void activemq::wireformat::openwire::utils::BooleanStream::marshal  
(std::vector< unsigned char > & dataOut)`

Marshal the data to a STL vector of unsigned chars.

### Parameters:

*dataOut* - reference to a vector to write the data to.

**6.140.3.3** `void activemq::wireformat::openwire::utils::BooleanStream::marshal  
(decaf::io::DataOutputStream * dataOut) throw ( decaf::io::IOException )`

Marshal the data to a DataOutputStream.

### Parameters:

*dataOut* - Stream to write the data to.

**6.140.3.4** `int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize  
( )`

Calc the size that data is marshalled to.

### Returns:

int size of marshalled data.

**6.140.3.5** `bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean  
( ) throw ( decaf::io::IOException )`

Read a boolean data element from the internal data buffer.

### Returns:

boolean from the stream

**6.140.3.6**    `void activemq::wireformat::openwire::utils::BooleanStream::unmarshal  
                  (decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )`

Unmarshal a Boolean data stream from the Input Stream.

**Parameters:**

*dataIn* - Input Stream to read data from.

**6.140.3.7**    `void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean  
                  (bool value) throw ( decaf::io::IOException )`

Writes a Boolean value to the internal data buffer.

**Parameters:**

*value* - boolean data to write.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/BooleanStream.h`

## 6.141 decaf::util::concurrent::BrokenBarrierException Class Reference

#include <src/main/decaf/util/concurrent/BrokenBarrierException.h> Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

### Public Member Functions

- **BrokenBarrierException** () throw ()  
*Default Constructor.*
- **BrokenBarrierException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BrokenBarrierException** (const **BrokenBarrierException** &ex) throw ()  
*Copy Constructor.*
- **BrokenBarrierException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BrokenBarrierException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BrokenBarrierException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BrokenBarrierException** \* clone () const  
*Clones this exception.*
- virtual ~**BrokenBarrierException** () throw ()

### 6.141.1 Constructor & Destructor Documentation

#### 6.141.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () [inline]

Default Constructor.

#### 6.141.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

#### 6.141.1.3 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException` (`const BrokenBarrierException & ex`) `throw ()` [inline]

Copy Constructor.

##### Parameters:

*ex* The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

#### 6.141.1.4 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException` (`const std::exception * cause`) `throw ()` [inline]

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.141.1.5 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException` (`const char * file`, `const int lineNumber`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs  
*lineNumber* - The line number where the exception occurred.  
*msg* - The message to report  
 ... - list of primitives that are formatted into the message

#### 6.141.1.6 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException` (`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs  
*lineNumber* - The line number where the exception occurred.  
*cause* - The exception that was the cause for this one to be thrown.  
*msg* - The message to report  
 ... - list of primitives that are formatted into the message

**6.141.1.7**    **virtual**  
**decaf::util::concurrent::BrokenBarrierException::~~BrokenBarrierException**  
**() throw ()**    [inline, virtual]

## 6.141.2 Member Function Documentation

**6.141.2.1**    **virtual BrokenBarrierException\*** **de-**  
**caf::util::concurrent::BrokenBarrierException::clone** ()  
**const**    [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BrokenBarrierException.h`

## 6.142 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

#include <src/main/activemq/commands/BrokerError.h> Inheritance diagram for activemq::commands::BrokerError:

### Data Structures

- struct **StackTraceElement**

### Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1660) Type as defined in **CommandTypes.h**.*
- virtual **BrokerError** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( **exceptions::ActiveMQException** )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- virtual const std::string & **getMessage** () const  
*Gets the string holding the error message.*
- virtual void **setMessage** (const std::string &message)  
*Sets the string that contains the error **Message** (p. 2516).*
- virtual const std::string & **getExceptionClass** () const  
*Gets the string holding the Exception Class name.*
- virtual void **setExceptionClass** (const std::string &exceptionClass)  
*Sets the string that contains the Exception Class name.*
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const  
*Gets the Broker Error that caused this exception.*
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)  
*Sets the Broker Error that caused this exception.*

- virtual const std::vector< **decaf::lang::Pointer< StackTraceElement > >** & **getStackTraceElements** () const  
*Gets the Stack Trace Elements for the Exception.*
- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer< StackTraceElement > >** & stackTraceElements)  
*Sets the Stack Trace Elements for this Exception.*

### 6.142.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

### 6.142.2 Constructor & Destructor Documentation

**6.142.2.1** **activemq::commands::BrokerError::BrokerError ()**

**6.142.2.2** **virtual activemq::commands::BrokerError::~~BrokerError ()** [virtual]

### 6.142.3 Member Function Documentation

**6.142.3.1** **virtual BrokerError\* activemq::commands::BrokerError::cloneDataStructure ()**  
const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

References `copyDataStructure()`.

**6.142.3.2** **virtual void activemq::commands::BrokerError::copyDataStructure (const DataStructure \* src)** [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

Referenced by `cloneDataStructure()`.

**6.142.3.3** `virtual const decaf::lang::Pointer<BrokerError>& activemq::commands::BrokerError::getCause () const [inline, virtual]`

Gets the Broker Error that caused this exception.

**Returns:**

Broker Error Pointer

**6.142.3.4** `virtual unsigned char activemq::commands::BrokerError::getDataStructureType () const [inline, virtual]`

Get the **DataStructure** (p.1660) Type as defined in CommandTypes.h.

**Returns:**

The type of the data structure

Implements **activemq::commands::DataStructure** (p.1662).

**6.142.3.5** `virtual const std::string& activemq::commands::BrokerError::getExceptionClass () const [inline, virtual]`

Gets the string holding the Exception Class name.

**Returns:**

Exception Class name

**6.142.3.6** `virtual const std::string& activemq::commands::BrokerError::getMessage () const [inline, virtual]`

Gets the string holding the error message.

**Returns:**

String Message (p.2516)

**6.142.3.7** `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >& activemq::commands::BrokerError::getStackTraceElements () const [inline, virtual]`

Gets the Stack Trace Elements for the Exception.

**Returns:**

Stack Trace Elements



**6.142.3.8** virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & *cause*) [inline, virtual]

Sets the Broker Error that caused this exception.

**Parameters:**

*cause* - Broker Error

**6.142.3.9** virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & *exceptionClass*) [inline, virtual]

Sets the string that contains the Exception Class name.

**Parameters:**

*exceptionClass* - String Exception Class name

**6.142.3.10** virtual void activemq::commands::BrokerError::setMessage (const std::string & *message*) [inline, virtual]

Sets the string that contains the error **Message** (p.2516).

**Parameters:**

*message* - String Error **Message** (p.2516)

**6.142.3.11** virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & *stackTraceElements*) [inline, virtual]

Sets the Stack Trace Elements for this Exception.

**Parameters:**

*stackTraceElements* - Stack Trace Elements

**6.142.3.12** virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor \* *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p.3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

## 6.143 activemq::exceptions::BrokerException Class Reference

#include <src/main/activemq/exceptions/BrokerException.h> Inheritance diagram for activemq::exceptions::BrokerException:

### Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const exceptions::ActiveMQException &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const commands::BrokerError \*error) throw ()
- virtual **BrokerException** \* clone () const  
*Clones this exception.*
- virtual ~**BrokerException** () throw ()

### 6.143.1 Constructor & Destructor Documentation

- 6.143.1.1 **activemq::exceptions::BrokerException::BrokerException () throw ()**  
[inline]
- 6.143.1.2 **activemq::exceptions::BrokerException::BrokerException (const exceptions::ActiveMQException & ex) throw ()** [inline]
- 6.143.1.3 **activemq::exceptions::BrokerException::BrokerException (const BrokerException & ex) throw ()** [inline]
- 6.143.1.4 **activemq::exceptions::BrokerException::BrokerException (const char \* file, const int lineNumber, const char \* msg, ...) throw ()** [inline]
- 6.143.1.5 **activemq::exceptions::BrokerException::BrokerException (const char \* file, const int lineNumber, const commands::BrokerError \* error) throw ()** [inline]

References decaf::lang::Exception::setMark(), and decaf::lang::Exception::setMessage().

- 6.143.1.6 **virtual activemq::exceptions::BrokerException::~~BrokerException () throw ()** [inline, virtual]

### 6.143.2 Member Function Documentation

- 6.143.2.1 **virtual BrokerException\* activemq::exceptions::BrokerException::clone () const** [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `activemq::exceptions::ActiveMQException` (p. 358).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

## 6.144 activemq::commands::BrokerId Class Reference

#include <src/main/activemq/commands/BrokerId.h> Inheritance diagram for activemq::commands::BrokerId:

### Public Types

- typedef decaf::lang::PointerComparator< BrokerId > COMPARATOR

### Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual ~**BrokerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

### Static Public Attributes

- static const unsigned char **ID\_BROKERID** = 124

### Protected Attributes

- std::string value

### 6.144.1 Member Typedef Documentation

**6.144.1.1** `typedef decaf::lang::PointerComparator<BrokerId>  
activemq::commands::BrokerId::COMPARATOR`

### 6.144.2 Constructor & Destructor Documentation

**6.144.2.1** `activemq::commands::BrokerId::BrokerId ()`

**6.144.2.2** `activemq::commands::BrokerId::BrokerId (const BrokerId & other)`

**6.144.2.3** `virtual activemq::commands::BrokerId::~~BrokerId () [virtual]`

### 6.144.3 Member Function Documentation

**6.144.3.1** `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ()  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.144.3.2** `virtual int activemq::commands::BrokerId::compareTo (const BrokerId &  
value) const [virtual]`

**6.144.3.3** `virtual void activemq::commands::BrokerId::copyDataStructure (const  
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

**6.144.3.4** `virtual bool activemq::commands::BrokerId::equals (const BrokerId &  
value) const [virtual]`

**6.144.3.5** `virtual bool activemq::commands::BrokerId::equals (const DataStructure  
* value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

**6.144.3.6** `virtual unsigned char activemq::commands::BrokerId::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

**6.144.3.7** `virtual std::string& activemq::commands::BrokerId::getValue () [virtual]`

**6.144.3.8** `virtual const std::string& activemq::commands::BrokerId::getValue () const [virtual]`

**6.144.3.9** `virtual bool activemq::commands::BrokerId::operator< (const BrokerId & value) const [virtual]`

**6.144.3.10** `BrokerId& activemq::commands::BrokerId::operator= (const BrokerId & other)`

**6.144.3.11** `virtual bool activemq::commands::BrokerId::operator== (const BrokerId & value) const [virtual]`

**6.144.3.12** `virtual void activemq::commands::BrokerId::setValue (const std::string & value) [virtual]`

**6.144.3.13** `virtual std::string activemq::commands::BrokerId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 833).

## 6.144.4 Field Documentation

**6.144.4.1** `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124 [static]`

**6.144.4.2** `std::string activemq::commands::BrokerId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

## 6.145 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 872).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.145.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 872). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.145.2 Constructor & Destructor Documentation

**6.145.2.1** `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.145.2.2** `virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.145.3 Member Function Documentation

**6.145.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.145.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.145.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.145.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.145.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.145.3.6** virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.145.3.7** virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h**

## 6.146 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 876).

`#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller`:

### Public Member Functions

- `BrokerIdMarshaller ()`
- virtual `~BrokerIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.146.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 876). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.146.2 Constructor & Destructor Documentation

**6.146.2.1** `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.146.2.2** `virtual activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.146.3 Member Function Documentation

**6.146.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.146.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.146.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.146.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.146.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.146.3.6** virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.146.3.7** virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h**

## 6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 880).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.147.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 880). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.147.2 Constructor & Destructor Documentation

**6.147.2.1** `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.147.2.2** `virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.147.3 Member Function Documentation

**6.147.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.147.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.147.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.147.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.147.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.147.3.6** virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.147.3.7** virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h**

## 6.148 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 884).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.148.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 884). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.148.2 Constructor & Destructor Documentation

**6.148.2.1** `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.148.2.2** `virtual activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.148.3 Member Function Documentation

**6.148.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.148.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.148.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.148.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.148.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.148.3.6** virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.148.3.7** virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerIdMarshaller.h**

## 6.149 `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 888).

`#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller`:

### Public Member Functions

- `BrokerIdMarshaller ()`
- virtual `~BrokerIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.149.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 888). **NOTE!**: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.149.2 Constructor & Destructor Documentation

**6.149.2.1** `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.149.2.2** `virtual activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.149.3 Member Function Documentation

**6.149.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.149.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.149.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.149.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.149.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.149.3.6** virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.149.3.7** virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**BrokerIdMarshaller.h**

## 6.150 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 892).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller`:

### Public Member Functions

- `BrokerIdMarshaller ()`
- `virtual ~BrokerIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.150.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `BrokerIdMarshaller` (p. 892). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.150.2 Constructor & Destructor Documentation

**6.150.2.1** `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

**6.150.2.2** `virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

## 6.150.3 Member Function Documentation

**6.150.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.150.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.150.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.150.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.150.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.150.3.6** virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.150.3.7** virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h**

## 6.151 activemq::commands::BrokerInfo Class Reference

#include <src/main/activemq/commands/BrokerInfo.h> Inheritance diagram for activemq::commands::BrokerInfo:

### Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool faultTolerantConfiguration)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool duplexConnection)

- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool **networkConnection**)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long **connectionId**)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &**brokerUploadUrl**)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &**networkProperties**)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_BROKERINFO** = 2

## Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< decaf::lang::Pointer< BrokerInfo > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

## 6.151.1 Constructor & Destructor Documentation

6.151.1.1 **activemq::commands::BrokerInfo::BrokerInfo** ()

6.151.1.2 **virtual activemq::commands::BrokerInfo::~~BrokerInfo** () [virtual]

## 6.151.2 Member Function Documentation

6.151.2.1 **virtual BrokerInfo\* activemq::commands::BrokerInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.



**Returns:**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.151.2.2 virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure \* *src*) [virtual]**

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.151.2.3 virtual bool activemq::commands::BrokerInfo::equals (const DataStructure \* *value*) const [virtual]**

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

- 6.151.2.4 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ()`  
[virtual]
- 6.151.2.5 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () const`  
[virtual]
- 6.151.2.6 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ()`  
[virtual]
- 6.151.2.7 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerName () const`  
[virtual]
- 6.151.2.8 `virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ()`  
[virtual]
- 6.151.2.9 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () const` [virtual]
- 6.151.2.10 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ()`  
[virtual]
- 6.151.2.11 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL () const`  
[virtual]
- 6.151.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId () const` [virtual]
- 6.151.2.13 `virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.151.2.14 `virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties ()`  
[virtual]
- 6.151.2.15 `virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties ()`  
`const` [virtual]
- 6.151.2.16 `virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ()` [virtual]
- 6.151.2.17 `virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ()` `const`  
[virtual]
- 6.151.2.18 `virtual bool activemq::commands::BrokerInfo::isBrokerInfo ()` `const`  
[inline, virtual]

**Returns:**

an answer of true to the `isBrokerInfo()` (p. 900) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 760).

- 6.151.2.19 virtual bool activemq::commands::BrokerInfo::isDuplexConnection () const [virtual]
- 6.151.2.20 virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration () const [virtual]
- 6.151.2.21 virtual bool activemq::commands::BrokerInfo::isMasterBroker () const [virtual]
- 6.151.2.22 virtual bool activemq::commands::BrokerInfo::isNetworkConnection () const [virtual]
- 6.151.2.23 virtual bool activemq::commands::BrokerInfo::isSlaveBroker () const [virtual]
- 6.151.2.24 virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & *brokerId*) [virtual]
- 6.151.2.25 virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & *brokerName*) [virtual]
- 6.151.2.26 virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & *brokerUploadUrl*) [virtual]
- 6.151.2.27 virtual void activemq::commands::BrokerInfo::setBrokerURL (const std::string & *brokerURL*) [virtual]
- 6.151.2.28 virtual void activemq::commands::BrokerInfo::setConnectionId (long long *connectionId*) [virtual]
- 6.151.2.29 virtual void activemq::commands::BrokerInfo::setDuplexConnection (bool *duplexConnection*) [virtual]
- 6.151.2.30 virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration (bool *faultTolerantConfiguration*) [virtual]
- 6.151.2.31 virtual void activemq::commands::BrokerInfo::setMasterBroker (bool *masterBroker*) [virtual]
- 6.151.2.32 virtual void activemq::commands::BrokerInfo::setNetworkConnection (bool *networkConnection*) [virtual]
- 6.151.2.33 virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & *networkProperties*) [virtual]
- 6.151.2.34 virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & *peerBrokerInfos*) [virtual]
- 6.151.2.35 virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool *slaveBroker*) [virtual]
- 6.151.2.36 virtual std::string activemq::commands::BrokerInfo::toString () const  
[virtual]      Generated on Wed Jul 25 23:57:04 2012 for activemq-cpp-3.2.5 by Doxygen

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.151.2.37** `virtual Pointer<Command> activemq::commands::BrokerInfo::visit  
(activemq::state::CommandVisitor * visitor) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.151.3 Field Documentation

- 6.151.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId` [protected]
- 6.151.3.2 `std::string activemq::commands::BrokerInfo::brokerName` [protected]
- 6.151.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl` [protected]
- 6.151.3.4 `std::string activemq::commands::BrokerInfo::brokerURL` [protected]
- 6.151.3.5 `long long activemq::commands::BrokerInfo::connectionId` [protected]
- 6.151.3.6 `bool activemq::commands::BrokerInfo::duplexConnection` [protected]
- 6.151.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration` [protected]
- 6.151.3.8 `const unsigned char activemq::commands::BrokerInfo::ID_ -  
BROKERINFO = 2` [static]
- 6.151.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.151.3.10 `bool activemq::commands::BrokerInfo::networkConnection` [protected]
- 6.151.3.11 `std::string activemq::commands::BrokerInfo::networkProperties` [protected]
- 6.151.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> >  
activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.151.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

## 6.152 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 904).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.152.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 904). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.152.2 Constructor & Destructor Documentation

**6.152.2.1** `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.152.2.2** `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

## 6.152.3 Member Function Documentation

**6.152.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.152.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.152.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.152.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.152.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.152.3.6** virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.152.3.7** virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h**

## 6.153 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 908).

#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.153.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 908). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.153.2 Constructor & Destructor Documentation

**6.153.2.1** `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.153.2.2** `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

### 6.153.3 Member Function Documentation

**6.153.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

#### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.153.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.153.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.153.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.153.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.153.3.6** virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.153.3.7** virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h**

## 6.154 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 912).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.154.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 912). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.154.2 Constructor & Destructor Documentation

**6.154.2.1** `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.154.2.2** `virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

## 6.154.3 Member Function Documentation

**6.154.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.154.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.154.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

```
6.154.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

```
6.154.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.154.3.6** virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.154.3.7** virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h**

## 6.155 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 916).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.155.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 916). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.155.2 Constructor & Destructor Documentation

**6.155.2.1** `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.155.2.2** `virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

## 6.155.3 Member Function Documentation

**6.155.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.155.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.155.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.155.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.155.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.155.3.6** virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.155.3.7** virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerInfoMarshaller.h**

## 6.156 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 920).

#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.156.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 920). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.156.2 Constructor & Destructor Documentation

**6.156.2.1** `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.156.2.2** `virtual activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

## 6.156.3 Member Function Documentation

**6.156.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.156.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.156.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.156.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.156.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.156.3.6** virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.156.3.7** virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**BrokerInfoMarshaller.h**

## 6.157 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 924).

#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.157.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 924). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.157.2 Constructor & Destructor Documentation

**6.157.2.1** `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

**6.157.2.2** `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

## 6.157.3 Member Function Documentation

**6.157.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.157.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.157.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.157.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.157.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.157.3.6** virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.157.3.7** virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**

## 6.158 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

#include <src/main/decaf/nio/Buffer.h> Inheritance diagram for decaf::nio::Buffer:

### Public Member Functions

- **Buffer** (int capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual int **capacity** () const
- virtual int **position** () const
- virtual **Buffer** & **position** (int newPosition) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's position.*
- virtual int **limit** () const
- virtual **Buffer** & **limit** (int newLimit) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's limit.*
- virtual **Buffer** & **mark** ()  
*Sets this buffer's mark at its position.*
- virtual **Buffer** & **reset** () throw ( InvalidMarkException )  
*Resets this buffer's position to the previously-marked position.*
- virtual **Buffer** & **clear** ()  
*Clears this buffer.*
- virtual **Buffer** & **flip** ()  
*Flips this buffer.*
- virtual **Buffer** & **rewind** ()  
*Rewinds this buffer.*
- virtual int **remaining** () const  
*Returns the number of elements between the current position and the limit.*
- virtual bool **hasRemaining** () const  
*Tells whether there are any elements between the current position and the limit.*
- virtual bool **isReadOnly** () const =0  
*Tells whether or not this buffer is read-only.*

## Protected Attributes

- int **\_position**
- int **\_capacity**
- int **\_limit**
- int **\_mark**
- bool **\_markSet**

### 6.158.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: \* Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p.957) and a relative put operation throws a **BufferOverflowException** (p.954); in either case, no data is transferred. \* Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p.2135) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values:  $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:



**clear()** (p. 930) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

**flip()** (p. 931) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

**rewind()** (p. 933) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 3169) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

## 6.158.2 Constructor & Destructor Documentation

**6.158.2.1** `decaf::nio::Buffer::Buffer (int capacity)`

**6.158.2.2** `decaf::nio::Buffer::Buffer (const Buffer & other)`

**6.158.2.3** `virtual decaf::nio::Buffer::~~Buffer ()` [inline, virtual]

## 6.158.3 Member Function Documentation

**6.158.3.1** `virtual int decaf::nio::Buffer::capacity () const` [inline, virtual]

**Returns:**

this buffer's capacity.

**6.158.3.2** `virtual Buffer& decaf::nio::Buffer::clear ()` [virtual]

Clears this buffer. The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

**Returns:**

a reference to this buffer.

### 6.158.3.3 virtual Buffer& decaf::nio::Buffer::flip () [virtual]

Flips this buffer. The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header in.read(buf); // Read data into rest of buffer buf.flip(); //
Flip buffer out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

**Returns:**

a reference to this buffer.

### 6.158.3.4 virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]

Tells whether there are any elements between the current position and the limit.

**Returns:**

true if, and only if, there is at least one element remaining in this buffer.

### 6.158.3.5 virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1013), `decaf::internal::nio::CharArrayBuffer` (p. 1116), `decaf::internal::nio::DoubleArrayBuffer` (p. 1806), `decaf::internal::nio::FloatArrayBuffer` (p. 1922), `decaf::internal::nio::IntArrayBuffer` (p. 2063), `decaf::internal::nio::LongArrayBuffer` (p. 2441), `decaf::internal::nio::ShortArrayBuffer` (p. 3456), and `decaf::nio::ByteBuffer` (p. 1046).

### 6.158.3.6 virtual Buffer& decaf::nio::Buffer::limit (int *newLimit*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's limit. If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

**Parameters:**

*newLimit* The new limit value; must be no larger than this buffer's capacity.

**Returns:**

A reference to This buffer

**Exceptions:**

*IllegalArgumentException* if preconditions on the new pos don't hold.

**6.158.3.7** virtual int decaf::nio::Buffer::limit () const [inline, virtual]

**Returns:**

this buffers Limit

**6.158.3.8** virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

**Returns:**

a reference to this buffer.

**6.158.3.9** virtual Buffer& decaf::nio::Buffer::position (int *newPosition*) throw (lang::exceptions::IllegalArgumentException ) [virtual]

Sets this buffer's position. If the mark is defined and larger than the new position then it is discarded.

**Parameters:**

*newPosition* The new postion in the buffer to set.

**Returns:**

a reference to This buffer.

**Exceptions:**

*IllegalArgumentException* if preconditions on the new pos don't hold.

**6.158.3.10** virtual int decaf::nio::Buffer::position () const [inline, virtual]

**Returns:**

the current position in the buffer

**6.158.3.11** `virtual int decaf::nio::Buffer::remaining () const` [inline, virtual]

Returns the number of elements between the current position and the limit.

**Returns:**

The number of elements remaining in this buffer

**6.158.3.12** `virtual Buffer& decaf::nio::Buffer::reset () throw ( InvalidMarkException )` [virtual]

Resets this buffer's position to the previously-marked position.

**Returns:**

a reference to this buffer.

**Exceptions:**

*InvalidMarkException* (p. 2135) - If the mark has not been set

**6.158.3.13** `virtual Buffer& decaf::nio::Buffer::rewind ()` [virtual]

Rewinds this buffer. The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array
```

**Returns:**

a reference to this buffer.

**6.158.4** Field Documentation**6.158.4.1** `int decaf::nio::Buffer::_capacity` [protected]**6.158.4.2** `int decaf::nio::Buffer::_limit` [protected]**6.158.4.3** `int decaf::nio::Buffer::_mark` [protected]**6.158.4.4** `bool decaf::nio::Buffer::_markSet` [protected]**6.158.4.5** `int decaf::nio::Buffer::_position` [mutable, protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/Buffer.h`

## 6.159 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of **io** (p. 154) operations on the input stream.

#include <src/main/decaf/io/BufferedInputStream.h> Inheritance diagram for decaf::io::BufferedInputStream:

### Public Member Functions

- **BufferedInputStream** (**InputStream** \*stream, bool **own**=false)  
*Constructor.*
- **BufferedInputStream** (**InputStream** \*stream, int bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException )  
*Constructor.*
- virtual ~**BufferedInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.*  
*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiaible. The caller should view the result of this method as an absolute.*  
*The default implementation of this method returns zero.*  
**Returns:**  
*the number of bytes available on this input stream.*  
**Exceptions:**  
*IOException (p. 2142) if an I/O error occurs.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.*  
*The default implementation of this method does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.*  
*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*  
**Parameters:**  
*num The number of bytes to skip.*  
**Returns:**  
*total bytes skipped*

**Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters:**

***readLimit** The max bytes read before marked position is invalid.*

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2142).*

**Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns:**

*true if this stream instance supports marks*

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

## 6.159.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of `io` (p. 154) operations on the input stream.

## 6.159.2 Constructor & Destructor Documentation

### 6.159.2.1 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, bool own = false)`

Constructor.

#### Parameters:

- stream* The target input stream to buffer.
- own* Indicates if we own the stream object, defaults to false.

### 6.159.2.2 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, int bufferSize, bool own = false) throw (lang::exceptions::IllegalArgumentException )`

Constructor.

#### Parameters:

- stream* The target input stream to buffer.
- bufferSize* The size in bytes to allocate for the **internal** (p. 144) buffer.
- own* Indicates if we own the stream object, defaults to false.

#### Exceptions:

- IllegalArgumentException* is the size is zero or negative.

### 6.159.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ()` [virtual]

## 6.159.3 Member Function Documentation

### 6.159.3.1 `virtual int decaf::io::BufferedInputStream::available () const throw (decaf::io::IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns:

- the number of bytes available on this input stream.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p.1896).

**6.159.3.2 virtual void decaf::io::BufferedInputStream::close () throw ( decaf::io::IOException ) [virtual]**

Closes the **InputStream** (p. 2043) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p.1897).

**6.159.3.3 virtual int decaf::io::BufferedInputStream::doReadArrayBounded (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]**

Reimplemented from **decaf::io::FilterInputStream** (p.1897).

**6.159.3.4 virtual int decaf::io::BufferedInputStream::doReadByte () throw ( decaf::io::IOException ) [protected, virtual]**

Reimplemented from **decaf::io::FilterInputStream** (p.1897).

**6.159.3.5 virtual void decaf::io::BufferedInputStream::mark (int *readLimit*) [virtual]**

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

**Parameters:**

*readLimit* The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p.1897).

**6.159.3.6 virtual bool decaf::io::BufferedInputStream::markSupported () const [inline, virtual]**

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.



The default implementation of this method returns false.

**Returns:**

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p.1898).

**6.159.3.7 virtual void decaf::io::BufferedInputStream::reset () throw ( decaf::io::IOException ) [virtual]**

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p.2142) might be thrown. \* If such an **IOException** (p.2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p.2142). \* If an **IOException** (p.2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p.2142).

**Exceptions:**

**IOException** (p.2142) if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p.1898).

**6.159.3.8 virtual long long decaf::io::BufferedInputStream::skip (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]**

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p.2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters:**

*num* The number of bytes to skip.

**Returns:**

total bytes skipped

**Exceptions:**

***IOException*** (p. 2142) if an I/O error occurs.

***UnsupportedOperationException*** if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p.1899).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedInputStream.h`

## 6.160 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

#include <src/main/decaf/io/BufferedOutputStream.h> Inheritance diagram for decaf::io::BufferedOutputStream:

### Public Member Functions

- **BufferedOutputStream** (OutputStream \*stream, bool **own**=false)  
*Constructor.*
- **BufferedOutputStream** (OutputStream \*stream, int bufferSize, bool **own**=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Constructor.*
- virtual ~**BufferedOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )  
*inheritDoc*
- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.160.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

### 6.160.2 Constructor & Destructor Documentation

#### 6.160.2.1 decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream \* *stream*, bool *own* = false)

Constructor.

#### Parameters:

***stream*** The target output stream.

***own*** Indicates if this class owns the stream pointer.

**6.160.2.2** `decaf::io::BufferedOutputStream::BufferedOutputStream`  
 (`OutputStream * stream`, `int bufferSize`, `bool own = false`) `throw (`  
`decaf::lang::exceptions::IllegalArgumentException )`

Constructor.

**Parameters:**

*stream* The target output stream.  
*bufferSize* The size for the **internal** (p.144) buffer.  
*own* Indicates if this class owns the stream pointer.

**Exceptions:**

*IllegalArgumentException* if the bufferSize given is negative.

**6.160.2.3** `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream ()`  
`[virtual]`

### 6.160.3 Member Function Documentation

**6.160.3.1** `virtual void decaf::io::BufferedOutputStream::doWriteArray` (`const`  
`unsigned char * buffer`, `int size`) `throw ( decaf::io::IOException )`  
`[protected, virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p.1902).

**6.160.3.2** `virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded`  
 (`const unsigned char * buffer`, `int size`, `int offset`, `int length`) `throw (`  
`decaf::io::IOException`, `decaf::lang::exceptions::NullPointerException`,  
`decaf::lang::exceptions::IndexOutOfBoundsException )` `[protected,`  
`virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p.1902).

**6.160.3.3** `virtual void decaf::io::BufferedOutputStream::doWriteByte` (`unsigned`  
`char c`) `throw ( decaf::io::IOException )` `[protected, virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p.1902).

**6.160.3.4** `virtual void decaf::io::BufferedOutputStream::flush ()` `throw (`  
`decaf::io::IOException )` `[virtual]`

`inheritDoc`}

Reimplemented from `decaf::io::FilterOutputStream` (p.1902).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedOutputStream.h`

## 6.161 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p.163) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

### Public Member Functions

- virtual `~BufferFactory()`

### Static Public Member Functions

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new ByteBuffer.*

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (std::vector< unsigned char > &buffer)

*Wraps the passed STL Byte Vector in a ByteBuffer.*

- static **decaf::nio::CharBuffer** \* **createCharBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::CharBuffer** \* **createCharBuffer** (char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new CharBuffer.*

- static **decaf::nio::CharBuffer** \* **createCharBuffer** (std::vector< char > &buffer)

*Wraps the passed STL Byte Vector in a CharBuffer.*

- static **decaf::nio::DoubleBuffer** \* **createDoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::DoubleBuffer** \* **createDoubleBuffer** (double \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new DoubleBuffer.*

- static **decaf::nio::DoubleBuffer** \* **createDoubleBuffer** (std::vector< double > &buffer)  
*Wraps the passed STL Double Vector in a DoubleBuffer.*
- static **decaf::nio::FloatBuffer** \* **createFloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::FloatBuffer** \* **createFloatBuffer** (float \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new FloatBuffer.*
- static **decaf::nio::FloatBuffer** \* **createFloatBuffer** (std::vector< float > &buffer)  
*Wraps the passed STL Float Vector in a FloatBuffer.*
- static **decaf::nio::LongBuffer** \* **createLongBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::LongBuffer** \* **createLongBuffer** (long long \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new LongBuffer.*
- static **decaf::nio::LongBuffer** \* **createLongBuffer** (std::vector< long long > &buffer)  
*Wraps the passed STL Long Vector in a LongBuffer.*
- static **decaf::nio::IntBuffer** \* **createIntBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::IntBuffer** \* **createIntBuffer** (int \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new IntBuffer.*
- static **decaf::nio::IntBuffer** \* **createIntBuffer** (std::vector< int > &buffer)  
*Wraps the passed STL int Vector in a IntBuffer.*
- static **decaf::nio::ShortBuffer** \* **createShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::ShortBuffer** \* **createShortBuffer** (short \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new ShortBuffer.*

- static **decaf::nio::ShortBuffer \* createShortBuffer** (std::vector< short > &buffer)  
*Wraps the passed STL Short Vector in a ShortBuffer.*

### 6.161.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p.163) package to create the various default version of the NIO interfaces.

**Since:**

1.0

### 6.161.2 Constructor & Destructor Documentation

**6.161.2.1** **virtual decaf::internal::nio::BufferFactory::~~BufferFactory ()** [inline, virtual]

### 6.161.3 Member Function Documentation

**6.161.3.1** **static decaf::nio::ByteBuffer\* decaf::internal::nio::BufferFactory::createByteBuffer** (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a ByteBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new ByteBuffer that is backed by buffer, caller owns.

**6.161.3.2** **static decaf::nio::ByteBuffer\* decaf::internal::nio::BufferFactory::createByteBuffer** (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Wraps the passed buffer with a new ByteBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.  
*size* The size of the specified buffer.  
*offset* The offset of the subarray to be used.  
*length* The length of the subarray to be used.

**Returns:**

a new ByteBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given is Null.  
*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.3** `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated ByteBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.4** `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & buffer) [static]`

Wraps the passed STL Byte Vector in a CharBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new CharBuffer that is backed by buffer, caller owns.



**6.161.3.5** static decaf::nio::CharBuffer\* decaf::internal::nio::BufferFactory::createCharBuffer (char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Wraps the passed buffer with a new CharBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.

*size* The size of the specified buffer.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new CharBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given is Null.

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.6** static decaf::nio::CharBuffer\* decaf::internal::nio::BufferFactory::createCharBuffer (int *capacity*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated CharBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.7** static decaf::nio::DoubleBuffer\* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & *buffer*) [static]

Wraps the passed STL Double Vector in a DoubleBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa.

The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

```
6.161.3.8 static decaf::nio::DoubleBuffer* de-
caf::internal::nio::BufferFactory::createDoubleBuffer
(double * buffer, int size, int offset, int length)
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]
```

Wraps the passed buffer with a new `DoubleBuffer`. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.

*size* The size of the specified buffer.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new `DoubleBuffer` that is backed by `buffer`, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given is Null.

*IndexOutOfBoundsException* if the capacity specified is negative.

```
6.161.3.9 static decaf::nio::DoubleBuffer* de-
caf::internal::nio::BufferFactory::createDoubleBuffer (int
capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
[static]
```

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated DoubleBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.10** static decaf::nio::FloatBuffer\* decaf::internal::nio::BufferFactory::createFloatBuffer  
(std::vector< float > & *buffer*) [static]

Wraps the passed STL Float Vector in a FloatBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new FloatBuffer that is backed by buffer, caller owns.

**6.161.3.11** static decaf::nio::FloatBuffer\* decaf::internal::nio::BufferFactory::createFloatBuffer  
(float \* *buffer*, int *size*, int *offset*, int *length*)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Wraps the passed buffer with a new FloatBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.

*size* The size of the specified buffer.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new FloatBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given in Null.

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.12** `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated FloatBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.13** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & buffer) [static]`

Wraps the passed STL int Vector in a IntBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new IntBuffer that is backed by *buffer*, caller owns.

**6.161.3.14** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int * buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new IntBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.

*size* The size of the specified buffer.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new IntBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given is Null.

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.15** static decaf::nio::IntBuffer\* decaf::internal::nio::BufferFactory::createIntBuffer (int *capacity*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated IntBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.16** static decaf::nio::LongBuffer\* decaf::internal::nio::BufferFactory::createLongBuffer (std::vector< long long > & *buffer*) [static]

Wraps the passed STL Long Vector in a LongBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new LongBuffer that is backed by buffer, caller owns.

```

6.161.3.17 static decaf::nio::LongBuffer* de-
caf::internal::nio::BufferFactory::createLongBuffer
(long long * buffer, int size, int offset, int length)
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

```

Wraps the passed buffer with a new LongBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.  
*size* The size of the specified buffer.  
*offset* The offset of the subarray to be used.  
*length* The length of the subarray to be used.

**Returns:**

a new LongBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given is Null.  
*IndexOutOfBoundsException* if the capacity specified is negative.

```

6.161.3.18 static decaf::nio::LongBuffer* de-
caf::internal::nio::BufferFactory::createLongBuffer (int
capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException
) [static]

```

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* - the **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated DoubleBuffer which the caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if the capacity specified is negative.

```

6.161.3.19 static decaf::nio::ShortBuffer* de-
caf::internal::nio::BufferFactory::createShortBuffer
(std::vector< short > & buffer) [static]

```

Wraps the passed STL Short Vector in a ShortBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The

new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

**6.161.3.20** `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new `ShortBuffer`. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The array that will back the new buffer.

*size* The size of the specified buffer.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new `ShortBuffer` that is backed by `buffer`, caller owns the returned pointer.

**Exceptions:**

*NullPointerException* if the buffer given in `Null`.

*IndexOutOfBoundsException* if the capacity specified is negative.

**6.161.3.21** `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters:**

*capacity* The **internal** (p. 144) buffer's capacity.

**Returns:**

a newly allocated ShortBuffer which the caller owns.

**Exceptions:**

***IndexOutOfBoundsException*** if the capacity specified is negative.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/BufferFactory.h`



## 6.162 decaf::nio::BufferOverflowException Class Reference

#include <src/main/decaf/nio/BufferOverflowException.h> Inheritance diagram for decaf::nio::BufferOverflowException:

### Public Member Functions

- **BufferOverflowException** () throw ()  
*Default Constructor.*
- **BufferOverflowException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const **BufferOverflowException** &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BufferOverflowException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BufferOverflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **BufferOverflowException** \* clone () const  
*Clones this exception.*
- virtual ~**BufferOverflowException** () throw ()

### 6.162.1 Constructor & Destructor Documentation

#### 6.162.1.1 decaf::nio::BufferOverflowException::BufferOverflowException () throw () [inline]

Default Constructor.

#### 6.162.1.2 decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

### 6.162.1.3 decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy, which is an instance of this type

### 6.162.1.4 decaf::nio::BufferOverflowException::BufferOverflowException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.162.1.5 decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.162.1.6 decaf::nio::BufferOverflowException::BufferOverflowException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor.

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.162.1.7** virtual decaf::nio::BufferOverflowException::~~BufferOverflowException  
( ) throw ( ) [inline, virtual]

## 6.162.2 Member Function Documentation

**6.162.2.1** virtual BufferOverflowException\* decaf::nio::BufferOverflowException::clone ( ) const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferOverflowException.h**

## 6.163 decaf::nio::BufferUnderflowException Class Reference

#include <src/main/decaf/nio/BufferUnderflowException.h> Inheritance diagram for decaf::nio::BufferUnderflowException:

### Public Member Functions

- **BufferUnderflowException** () throw ()  
*Default Constructor.*
- **BufferUnderflowException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **BufferUnderflowException** (const BufferUnderflowException &ex) throw ()  
*Copy Constructor.*
- **BufferUnderflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BufferUnderflowException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BufferUnderflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **BufferUnderflowException** \* clone () const  
*Clones this exception.*
- virtual ~**BufferUnderflowException** () throw ()

### 6.163.1 Constructor & Destructor Documentation

**6.163.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException () throw () [inline]**

Default Constructor.

**6.163.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & ex) throw () [inline]**

Copy Constructor.

#### Parameters:

*ex* the exception to copy

### 6.163.1.3 decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy, which is an instance of this type

### 6.163.1.4 decaf::nio::BufferUnderflowException::BufferUnderflowException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.163.1.5 decaf::nio::BufferUnderflowException::BufferUnderflowException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.163.1.6 decaf::nio::BufferUnderflowException::BufferUnderflowException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor.

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.163.1.7**   **virtual**  
**decaf::nio::BufferUnderflowException::~~BufferUnderflowException ()**  
**throw ()**   [inline, virtual]

## **6.163.2   Member Function Documentation**

**6.163.2.1**   **virtual BufferUnderflowException\* de-**  
**caf::nio::BufferUnderflowException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferUnderflowException.h`

## 6.164 decaf::lang::Byte Class Reference

#include <src/main/decaf/lang/Byte.h> Inheritance diagram for decaf::lang::Byte:

### Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const  
*Compares this **Byte** (p. 960) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Byte** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const unsigned char &c) const  
*Compares this **Byte** (p. 960) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const unsigned char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3665) into a **Byte** (p. 960).*
- static unsigned char **parseByte** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed unsigned char in the radix specified by the second argument.*
- static unsigned char **parseByte** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal unsigned char.*
- static **Byte valueOf** (unsigned char value)  
*Returns a **Character** (p. 1100) instance representing the specified char value.*
- static **Byte valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Byte** (p. 960) object holding the value given by the specified std::string.*
- static **Byte valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Byte** (p. 960) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

## Static Public Attributes

- static const unsigned char **MIN\_VALUE** = 0x7F  
*The minimum value that a unsigned char can take on.*
- static const unsigned char **MAX\_VALUE** = 0x80  
*The maximum value that a unsigned char can take on.*
- static const int **SIZE** = 8  
*The size of the primitive character in bits.*

## 6.164.1 Constructor & Destructor Documentation

### 6.164.1.1 decaf::lang::Byte::Byte (unsigned char *value*)

#### Parameters:

*value* - the primitive value to wrap



### 6.164.1.2 decaf::lang::Byte::Byte (const std::string & *value*) throw ( exceptions::NumberFormatException )

#### Parameters:

*value* - the string to convert to an unsigned char

#### Exceptions:

*NumberFormatException*

### 6.164.1.3 virtual decaf::lang::Byte::~~Byte () [inline, virtual]

## 6.164.2 Member Function Documentation

### 6.164.2.1 virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2835).

### 6.164.2.2 virtual int decaf::lang::Byte::compareTo (const unsigned char & *c*) const [inline, virtual]

Compares this **Byte** (p. 960) instance with a char type.

#### Parameters:

*c* - the char instance to be compared

#### Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1214).

### 6.164.2.3 virtual int decaf::lang::Byte::compareTo (const Byte & *c*) const [inline, virtual]

Compares this **Byte** (p. 960) instance with another.

#### Parameters:

*c* - the **Byte** (p. 960) instance to be compared

**Returns:**

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

#### 6.164.2.4 static `Byte decaf::lang::Byte::decode (const std::string & value) throw ( exceptions::NumberFormatException )` [static]

Decodes a **String** (p. 3665) into a **Byte** (p. 960). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 966) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3665) is the minus sign. No whitespace characters are permitted in the string.

**Parameters:**

*value* - The string to decode

**Returns:**

a **Byte** (p. 960) object containing the decoded value

**Exceptions:**

***NumberFormatException*** if the string is not formatted correctly.

#### 6.164.2.5 virtual double `decaf::lang::Byte::doubleValue () const` [inline, virtual]

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

#### 6.164.2.6 bool `decaf::lang::Byte::equals (const unsigned char & c) const` [inline, virtual]

**Returns:**

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1215).

#### 6.164.2.7 bool `decaf::lang::Byte::equals (const Byte & c) const` [inline]

**Returns:**

true if the two **Byte** (p. 960) Objects have the same value.

**6.164.2.8 virtual float decaf::lang::Byte::floatValue () const [inline, virtual]**

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.164.2.9 virtual int decaf::lang::Byte::intValue () const [inline, virtual]**

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.164.2.10 virtual long long decaf::lang::Byte::longValue () const [inline, virtual]**

Answers the long value which the receiver represents.

**Returns:**

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.164.2.11 virtual bool decaf::lang::Byte::operator< (const unsigned char & c) const [inline, virtual]**

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1215).

**6.164.2.12 virtual bool decaf::lang::Byte::operator< (const Byte & c) const [inline, virtual]**

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.164.2.13** **virtual bool decaf::lang::Byte::operator== (const unsigned char & *c*)**  
**const** [inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1215).

**6.164.2.14** **virtual bool decaf::lang::Byte::operator== (const Byte & *c*) const**  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.164.2.15** **static unsigned char decaf::lang::Byte::parseByte (const std::string & *s*)**  
**throw ( exceptions::NumberFormatException )** [static]

Parses the string argument as a signed decimal unsigned char. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseByte( const std::string, int )` method.

**Parameters:**

*s* - **String** (p. 3665) to convert to a unsigned char

**Returns:**

the converted unsigned char value

**Exceptions:**

*NumberFormatException* if the string is not a unsigned char.

**6.164.2.16** static unsigned char decaf::lang::Byte::parseByte (const std::string & *s*, int *radix*) throw ( exceptions::NumberFormatException ) [static]

Parses the string argument as a signed unsigned char in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1103) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:

- \* The first argument is null or is a string of length zero.
- \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1107) or larger than **Character::MAX\_RADIX** (p. 1106).
- \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1.
- \* The value represented by the string is not a value of type unsigned char.

**Parameters:**

*s* - the **String** (p. 3665) containing the unsigned char to be parsed

*radix* - the radix to be used while parsing *s*

**Returns:**

the unsigned char represented by the string argument in the specified radix.

**Exceptions:**

*NumberFormatException* - If **String** (p. 3665) does not contain a parsable unsigned char.

**6.164.2.17** virtual short decaf::lang::Byte::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

**Returns:**

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2837).

**6.164.2.18** static std::string decaf::lang::Byte::toString (unsigned char *value*) [static]**Returns:**

a string representing the primitive value as Base 10

**6.164.2.19** std::string decaf::lang::Byte::toString () const**Returns:**

this **Byte** (p. 960) Object as a **String** (p. 3665) Representation

**6.164.2.20** `static Byte decaf::lang::Byte::valueOf (const std::string & value, int radix) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p.960) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte( std::string, int )` method. The result is a **Byte** (p.960) object that represents the unsigned char value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base ( *radix* )

*radix* - base of the string to parse.

**Returns:**

new **Byte** (p.960) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a valid unsigned char.

**6.164.2.21** `static Byte decaf::lang::Byte::valueOf (const std::string & value) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p.960) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte( std::string )` method. The result is a **Byte** (p.960) object that represents the unsigned char value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base 10

**Returns:**

new **Byte** (p.960) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a decimal unsigned char.

**6.164.2.22** `static Byte decaf::lang::Byte::valueOf (unsigned char value) [inline, static]`

Returns a **Character** (p.1100) instance representing the specified char value.

**Parameters:**

*value* - the primitive char to wrap.

**Returns:**

a new **Character** (p.1100) instance that wraps this value.

### 6.164.3 Field Documentation

**6.164.3.1** `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80` [static]

The maximum value that a unsigned char can take on.

**6.164.3.2** `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` [static]

The minimum value that a unsigned char can take on.

**6.164.3.3** `const int decaf::lang::Byte::SIZE = 8` [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

## 6.165 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

### Data Structures

- union **Array**

### Public Member Functions

- **ByteArrayAdapter** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.*

- **ByteArrayAdapter** (unsigned char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (double \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (float \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (long long \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (int \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte array object that wraps the given array.*



- **ByteArrayAdapter** (short \*array, int size, bool own=false)  
throw ( decaf::lang::exceptions::NullPointerException, de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- virtual ~**ByteArrayAdapter** ()
- virtual int **getCapacity** () const  
*Gets the size of the underlying array.*
- virtual int **getCharCapacity** () const  
*Gets the size of the underlying array as if it contains chars.*
- virtual int **getDoubleCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getFloatCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getLongCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getIntCapacity** () const  
*Gets the size of the underlying array as if it contains ints.*
- virtual int **getShortCapacity** () const  
*Gets the size of the underlying array as if it contains shorts.*
- virtual unsigned char \* **getByteArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual char \* **getCharArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual short \* **getShortArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual int \* **getIntArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual long long \* **getLongArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual double \* **getDoubleArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual float \* **getFloatArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual void **read** (unsigned char \*buffer, int size, int offset, int length)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, de-  
caf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException )

*Reads from the Byte array starting at the specified offset and reading the specified length.*

- virtual void **write** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException )

*Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects **internal** (p. 144) array.*

- virtual void **resize** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException )

*Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.*

- virtual void **clear** () throw ()

*Clear all data from that Array, setting the underlying bytes to zero.*

- unsigned char & **operator[]** (int index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allows the **ByteArrayAdapter** (p. 969) to be indexed as a standard array.*

- const unsigned char & **operator[]** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

- virtual unsigned char **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

- virtual char **getChar** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads one byte at the given index and returns it.*

- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

- virtual double **getDoubleAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given byte index and returns it.*

- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

- virtual float **getFloatAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given byte index and returns it.*

- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

- virtual long long **getLongAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given byte index and returns it.*
- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual int **getIntAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given byte index and returns it.*
- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given index and returns it.*
- virtual short **getShortAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given byte index and returns it.*
- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes the given byte into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes one byte containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given byte index.*

- virtual **ByteArrayAdapter** & **putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes two bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes two bytes containing the given value, into this buffer at the given byte index.*

### 6.165.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

**Since:**

1.0

### 6.165.2 Constructor & Destructor Documentation

#### 6.165.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int size) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

*size* The size of the array, this is the limit we read and write to.

**Exceptions:**

*IllegalArgumentException* if size is negative.

#### 6.165.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char \* array, int size, bool own = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

- array* The physical array to wrap.
- size* The size of the array, this is the limit we read and write to.
- own* Indicates if this class is now the owner of the pointer.

**Exceptions:**

- NullPointerException* if buffer is NULL
- IndexOutOfBoundsException* if the size is negative.

**6.165.2.3** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`  
(char \* *array*, int *size*, bool *own* = false) throw  
( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

- array* The physical array to wrap.
- size* The size of the array, this is the limit we read and write to.
- own* Indicates if this class is now the owner of the pointer.

**Exceptions:**

- NullPointerException* if buffer is NULL
- IndexOutOfBoundsException* if the size is negative.

**6.165.2.4** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`  
(double \* *array*, int *size*, bool *own* = false)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

- array* The physical array to wrap.
- size* The size of the array, this is the limit we read and write to.
- own* Indicates if this class is now the owner of the pointer.

**Exceptions:**

- NullPointerException* if buffer is NULL
- IndexOutOfBoundsException* if the size is negative.

**6.165.2.5** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`  
 (float \* *array*, int *size*, bool *own* = false) throw  
 ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The physical array to wrap.  
*size* The size of the array, this is the limit we read and write to.  
*own* Indicates if this class is now the owner of the pointer.

**Exceptions:**

*NullPointerException* if buffer is NULL  
*IndexOutOfBoundsException* if the size is negative.

**6.165.2.6** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`  
 (long long \* *array*, int *size*, bool *own* = false)  
 throw ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The physical array to wrap.  
*size* The size of the array, this is the limit we read and write to.  
*own* Indicates if this class is now the owner of the pointer.

**Exceptions:**

*NullPointerException* if buffer is NULL  
*IndexOutOfBoundsException* if the size is negative.

**6.165.2.7** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter`  
 (int \* *array*, int *size*, bool *own* = false) throw  
 ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The physical array to wrap.  
*size* The size of the array, this is the limit we read and write to.

*own* Indicates if this class is now the owner of the pointer.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if the size is negative.

**6.165.2.8** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter  
(short * array, int size, bool own = false)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

#### Parameters:

*array* The physical array to wrap.

*size* The size of the array, this is the limit we read and write to.

*own* Indicates if this class is now the owner of the pointer.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if the size is negative.

**6.165.2.9** `virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ()  
[virtual]`

### 6.165.3 Member Function Documentation

**6.165.3.1** `virtual void decaf::internal::util::ByteArrayAdapter::clear () throw ()  
[virtual]`

Clear all data from that Array, setting the underlying bytes to zero.

**6.165.3.2** `virtual unsigned char decaf::internal::util::ByteArrayAdapter::get (int  
index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
) [virtual]`

Absolute get method. Reads the byte at the given index.

#### Parameters:

*index* The index in the Buffer where the byte is to be read.

#### Returns:

the byte that is located at the given index.

#### Exceptions:

*IndexOutOfBoundsException* If index is not smaller than the buffer's limit or is negative.

**6.165.3.3** `virtual unsigned char* decaf::internal::util::ByteArrayAdapter::getByteArray ()`  
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.969) objects that point to this array.

**Returns:**

an unsigned char\* pointer to the array this object wraps.

**6.165.3.4** `virtual int decaf::internal::util::ByteArrayAdapter::getCapacity () const`  
[inline, virtual]

Gets the size of the underlying array.

**Returns:**

the size the array.

**6.165.3.5** `virtual char decaf::internal::util::ByteArrayAdapter::getChar (int index)`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads one byte at the given index and returns it.

**Parameters:**

*index* The index in the Buffer where the byte is to be read.

**Returns:**

the byte that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* If index is not smaller than the buffer's limit or is negative.

**6.165.3.6** `virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ()`  
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.969) objects that point to this array.

**Returns:**

an char\* pointer to the array this object wraps.

**6.165.3.7** `virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ()`  
`const` [inline, virtual]

Gets the size of the underlying array as if it contains chars.



**Returns:**

the size the array.

**6.165.3.8** virtual double decaf::internal::util::ByteArrayAdapter::getDouble (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read.

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.9** virtual double\* decaf::internal::util::ByteArrayAdapter::getDoubleArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.969) objects that point to this array.

**Returns:**

an double\* pointer to the array this object wraps.

**6.165.3.10** virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads eight bytes at the given byte index and returns it.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.11** `virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity  
( ) const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns:**

the size the array.

**6.165.3.12** `virtual float decaf::internal::util::ByteArrayAdapter::getFloat (int index)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read.

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.13** `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ( )  
[inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 969) objects that point to this array.

**Returns:**

an float\* pointer to the array this object wraps.

**6.165.3.14** `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (int in-  
dex) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
) [virtual]`

Reads four bytes at the given byte index and returns it.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.15**    `virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity ()  
                 const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns:**

the size the array.

**6.165.3.16**    `virtual int decaf::internal::util::ByteArrayAdapter::getInt (int index)  
                 const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
                 [virtual]`

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read.

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.17**    `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray ()  
                 [inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 969) objects that point to this array.

**Returns:**

an int\* pointer to the array this object wraps.

**6.165.3.18**    `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (int index)  
                 const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
                 [virtual]`

Reads four bytes at the given byte index and returns it.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.19** `virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity ()  
const [inline, virtual]`

Gets the size of the underlying array as if it contains ints.

**Returns:**

the size the array.

**6.165.3.20** `virtual long long decaf::internal::util::ByteArrayAdapter::getLong  
(int index) const throw ( de-  
cafe::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read.

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.21** `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray  
() [inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.969) objects that point to this array.

**Returns:**

an long long\* pointer to the array this object wraps.

**6.165.3.22** `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads eight bytes at the given byte index and returns it.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.23** `virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity ()`  
`const [inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns:**

the size the array.

**6.165.3.24** `virtual short decaf::internal::util::ByteArrayAdapter::getShort (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads two bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read.

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.25** `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray ()`  
`[inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.969) objects that point to this array.

**Returns:**

an short\* pointer to the array this object wraps.

**6.165.3.26** `virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Reads two bytes at the given byte index and returns it.

**Parameters:**

*index* The index in the Buffer where the bytes are to be read

**Returns:**

the value at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

**6.165.3.27** `virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity ()`  
`const [inline, virtual]`

Gets the size of the underlying array as if it contains shorts.

**Returns:**

the size the array.

**6.165.3.28** `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

**6.165.3.29** `unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (int index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

Allows the **ByteArrayAdapter** (p.969) to be indexed as a standard array. calling the non constant version allows the user to change the value at index

**Parameters:**

*index* The position in the array to access, if the value is negative or greater than the size of the underlying array an *IndexOutOfBoundsException* is thrown.

**Exceptions:**

*IndexOutOfBoundsException* if the preconditions of index are not met.

**6.165.3.30** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put (int *index*, unsigned char *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes the given byte into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.31** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar (int *index*, char *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes one byte containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.32** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.33** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.34** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.



**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.35** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (int *index*, float *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.36** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (int *index*, int *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.37** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.38** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.39** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.40** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (int *index*, short *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes two bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write to the array.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.41** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (int *index*, short *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

**Parameters:**

*index* The position in the Buffer to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

***IndexOutOfBoundsException*** if index greater than the buffer's limit minus the size of the type being written, or index is negative.

**6.165.3.42** `virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * buffer, int size, int offset, int length) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException )` [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length. If the length is greater than the size of this underlying byte array then an `BufferUnderflowException` is thrown.

**Parameters:**

*buffer* The buffer to read data from this array into.

*size* The size of the buffer passed.

*offset* The position in this array to start reading from.

*length* The amount of data to read from this array.

**Exceptions:**

*IndexOutOfBoundsException* if the offset + length exceeds the size.

*NullPointerException* if buffer is null

*BufferUnderflowException* if there is not enough data to read because the offset or the length is greater than the size of this array.

**6.165.3.43** `virtual void decaf::internal::util::ByteArrayAdapter::resize (int size) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException )` [virtual]

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved. A `ByteArrayAdapter` (p. 969) can only be resized when it owns the underlying array, if it does not then it will throw an `InvalidStateException`.

**Parameters:**

*size* The new size of the array.

**Exceptions:**

*IllegalArgumentException* if the size parameter is negative.

*InvalidStateException* if this object does not own the buffer.

**6.165.3.44** `virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException )` [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects `internal` (p. 144) array. . If the length is greater than the size of this underlying byte array then an `BufferOverflowException` is thrown.

**Parameters:**

- buffer* The buffer to read data from this array into.
- size* The size of the buffer passed.
- offset* The position in this array to start reading from.
- length* The amount of data to read from this array.

**Exceptions:**

- IndexOutOfBoundsException* if the offset + length exceeds the size.
- NullPointerException* if buffer is null
- BufferOverflowException* if the amount of data to be written to this array or the offset given are larger than this array's size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/ByteArrayAdapter.h`

## 6.166 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

#include <src/main/decaf/internal/nio/ByteBuffer.h> Inheritance diagram for decaf::internal::nio::ByteBuffer:

### Public Member Functions

- **ByteBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **ByteBuffer** (p. 991) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **ByteBuffer** (unsigned char \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **ByteBuffer** (p. 991) object that wraps the given array.*

- **ByteBuffer** (const decaf::lang::Pointer< **ByteBufferAdapter** > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed **ByteBufferAdapter** and start at the given offset.*

- **ByteBuffer** (const **ByteBuffer** &other)

*Create a **ByteBuffer** (p. 991) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferAdapter** and when changes are made to that data it is reflected in both.*

- virtual ~**ByteBuffer** ()
- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns:**

*true if, and only if, this buffer is read-only*

- virtual unsigned char \* **array** () throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )

*Returns the byte array that backs this buffer.*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The array that backs this buffer*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this buffer is backed by an array but is read-only  
**UnsupportedOperationException** if this buffer is not backed by an accessible array*

- virtual int **arrayOffset** () const throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position  $p$  corresponds to array index  $p + \text{arrayOffset}()$  (p. 1037).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset within this buffer's array of the first element of the buffer.

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is backed by an array but is read-only.  
**UnsupportedOperationException** if this buffer is not backed by an accessible array.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual decaf::nio::CharBuffer \* **asCharBuffer** () const

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new Char **Buffer** (p. 928), which the caller then owns.

- virtual decaf::nio::DoubleBuffer \* **asDoubleBuffer** () const

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new double **Buffer** (p. 928), which the caller then owns.

- virtual decaf::nio::FloatBuffer \* **asFloatBuffer** () const

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new float **Buffer** (p. 928), which the caller then owns.

- virtual **decaf::nio::IntBuffer \* asIntBuffer ()** const

*Creates a view of this byte buffer as a int buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the new int **Buffer** (p. 928), which the caller then owns.*

- virtual **decaf::nio::LongBuffer \* asLongBuffer ()** const

*Creates a view of this byte buffer as a long buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the new long **Buffer** (p. 928), which the caller then owns.*

- virtual **decaf::nio::ShortBuffer \* asShortBuffer ()** const

*Creates a view of this byte buffer as a short buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the new short **Buffer** (p. 928), which the caller then owns.*

- virtual **ByteBuffer \* asReadOnlyBuffer ()** const

*Creates a new, read-only byte buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only byte buffer which the caller then owns.*

- virtual **ByteBuffer & compact ()** throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*



**Returns:**

a reference to this **ByteBuffer** (p. 1031).

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer \* duplicate ()**

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new **Byte Buffer** (p. 928) which the caller owns.

- virtual unsigned char **get ()** const throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

**Returns:**

The byte at the buffer's current position.

**Exceptions:**

**BufferUnderflowException** (p. 957) if the buffer's current position is not smaller than its limit.

- virtual unsigned char **get (int index)** const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the byte at the given index.

**Parameters:**

**index** The index in the **Buffer** (p. 928) where the byte is to be read.

**Returns:**

the byte that is located at the given index.

**Exceptions:**

**IndexOutOfBoundsException** if **index** is not smaller than the buffer's limit, or **index** is negative.

- virtual char **getChar ()** throw ( decaf::nio::BufferUnderflowException )

Reads the next byte at this buffer's current position, and then increments the position by one.

**Returns:**

the next char in the buffer.

**Exceptions:**

**BufferUnderflowException** (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual char **getChar (int index)** const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Reads one byte at the given index and returns it.

**Parameters:**

**index** The index in the **Buffer** (p. 928) where the byte is to be read.

**Returns:**

*the char at the given index in the buffer*

**Exceptions:**

**IndexOutOfBoundsException** if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual double **getDouble** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns:**

*the next double in the buffer.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters:**

**index** The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

*the double at the given index in the buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual float **getFloat** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns:**

*the next float in the buffer.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters:**

**index** The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

*the float at the given index in the buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if there are not enough bytes remaining to fill the requested Data Type, or *index* is negative.

- virtual long long **getLong** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns:**

*the next long long in the buffer.*

**Exceptions:**

**BufferUnderflowException** (p. 957) *if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters:**

**index** *The index in the **Buffer** (p. 928) where the bytes are to be read.*

**Returns:**

*the long long at the given index in the buffer.*

**Exceptions:**

**IndexOutOfBoundsException** *if there are not enough bytes remaining to fill the requested Data Type, or index is negative.*

- virtual int **getInt** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns:**

*the next int in the buffer.*

**Exceptions:**

**BufferUnderflowException** (p. 957) *if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters:**

**index** *The index in the **Buffer** (p. 928) where the bytes are to be read.*

**Returns:**

*the int at the given index in the buffer.*

**Exceptions:**

**IndexOutOfBoundsException** *if there are not enough bytes remaining to fill the requested Data Type, or index is negative.*

- virtual short **getShort** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*

**Returns:**

*the next short in the buffer.*

**Exceptions:**

**BufferUnderflowException** (p. 957) *if there are no more bytes remaining in this buffer, meaning we have reached the set limit.*

- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads two bytes at the given index and returns it.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

*the short at the given index in the buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

- virtual **ByteBuffer** & **put** (unsigned char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the current position, and then increments the position.*

**Parameters:**

*value* - the byte value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the given index.*

**Parameters:**

*index* - position in the **Buffer** (p. 928) to write the data  
*value* - the byte to write.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putChar** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*

**Parameters:**

*value* The value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual **ByteBuffer** & **putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the given index.*

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The value to write.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters:**

*value* The value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data  
*value* The value to write.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putFloat** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters:**

*value* The value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters:**

**index** The position in the **Buffer** (p. 928) to write the data  
**value** The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putLong** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

**value** The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters:**

**index** The position in the **Buffer** (p. 928) to write the data.  
**value** The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putInt** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

**value** The value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual **ByteBuffer** & **putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

**Parameters:**

**index** *The position in the **Buffer** (p. 928) to write the data.*  
**value** *The value to write.*

**Returns:**

*a reference to this buffer*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual **ByteBuffer** & **putShort** (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters:**

**value** *The value to be written.*

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** & **putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the given index.*

**Parameters:**

**index** *The position in the **Buffer** (p. 928) to write the data*  
**value** *The value to write.*

**Returns:**

*a reference to this buffer*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **ByteBuffer** \* **slice** () const

*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the newly create **ByteBuffer** (p. 1031) which the caller owns.*

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 991) as Read-Only or not Read-Only.*

### 6.166.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

**float getFloat()** (p. 1010) **float getFloat(int index)** void **putFloat(float f)** (p. 1016) void **putFloat(int index, float f)** (p. 1015)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The **asFloatBuffer** method, for example, creates an instance of the **FloatBuffer** class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:



A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

**Since:**

1.0

## 6.166.2 Constructor & Destructor Documentation

### 6.166.2.1 decaf::internal::nio::ByteBuffer::ByteBuffer (int *capacity*, bool *readOnly* = false) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **ByteBuffer** (p. 991) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

*capacity* The size of the array, this is the limit we read and write to.

*readOnly* Should this buffer be read-only, default as false

**Exceptions:**

*IllegalArgumentException* if the capacity value is negative.

### 6.166.2.2 decaf::internal::nio::ByteBuffer::ByteBuffer (unsigned char \* *array*, int *size*, int *offset*, int *length*, bool *readOnly* = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **ByteBuffer** (p. 991) object that wraps the given array.

**Parameters:**

*array* The array to wrap.

*size* The size of the array passed.

*offset* The position that is this buffers start position.

*length* The size of the sub-array, this is the limit we read and write to.

*readOnly* Should this buffer be read-only, default as false.

**Exceptions:**

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if the preconditions of size, offset and length are violated.

**6.166.2.3** `decaf::internal::nio::ByteBuffer::ByteBuffer (const decaf::lang::Pointer< ByteBufferAdapter > & array, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteBufferAdapter and start at the given offset. The capacity and limit of the new **ByteBuffer** (p. 991) will be that of the remaining capacity of the passed buffer.

**Parameters:**

*array* The ByteBufferAdapter to wrap  
*offset* The offset into array where the buffer starts  
*length* The length of the array we are wrapping or limit.  
*readOnly* Boolean indicating if this a readOnly buffer.

**Exceptions:**

*NullPointerException* if array is NULL  
*IndexOutOfBoundsException* if offset is greater than array capacity.

**6.166.2.4** `decaf::internal::nio::ByteBuffer::ByteBuffer (const ByteBuffer & other)`

Create a **ByteBuffer** (p. 991) that mirrors this one, meaning it shares a reference to this buffers ByteBufferAdapter and when changes are made to that data it is reflected in both.

**Parameters:**

*other* The **ByteBuffer** (p. 991) this one is to mirror.

**6.166.2.5** `virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer () [virtual]`

## 6.166.3 Member Function Documentation

**6.166.3.1** `virtual unsigned char* decaf::internal::nio::ByteBuffer::array () throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The array that backs this buffer

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is backed by an array but is read-only  
*UnsupportedOperationException* if this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p.1037).

**6.166.3.2** `virtual int decaf::internal::nio::ByteBuffer::arrayOffset  
 () const throw ( decaf::nio::ReadOnlyBufferException,  
 decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + **array-Offset()** (p.1037).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset within this buffer's array of the first element of the buffer.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is backed by an array but is read-only.  
*UnsupportedOperationException* if this buffer is not backed by an accessible array.

Implements **decaf::nio::ByteBuffer** (p.1037).

**6.166.3.3** `virtual decaf::nio::CharBuffer* de-  
 caf::internal::nio::ByteBuffer::asCharBuffer () const  
 [inline, virtual]`

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new **Char Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p.1037).

**6.166.3.4** `virtual decaf::nio::DoubleBuffer* de-  
 caf::internal::nio::ByteBuffer::asDoubleBuffer ()  
 const [inline, virtual]`

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new double **Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1038).

**6.166.3.5** `virtual decaf::nio::FloatBuffer* decaf::internal::nio::ByteBuffer::asFloatBuffer () const`  
[inline, virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new float **Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1038).

**6.166.3.6** `virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer () const`  
[inline, virtual]

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new int **Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1038).

**6.166.3.7** `virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteBuffer::asLongBuffer () const`  
[inline, virtual]

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new long **Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1039).

**6.166.3.8** `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer () const`  
[virtual]

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1039).

**6.166.3.9** `virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer () const`  
[inline, virtual]

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new short **Buffer** (p. 928), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1039).

**6.166.3.10** **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact () throw ( decaf::nio::ReadOnlyBufferException )** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **ByteBuffer** (p. 1031).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1040).

**6.166.3.11** **virtual ByteBuffer\* decaf::internal::nio::ByteBuffer::duplicate ()** [virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new **Byte Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1040).

**6.166.3.12** **virtual unsigned char decaf::internal::nio::ByteBuffer::get (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )** [virtual]

Absolute get method.

Reads the byte at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the byte is to be read.

**Returns:**

the byte that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1040).

**6.166.3.13** `virtual unsigned char decaf::internal::nio::ByteBuffer::get () const  
throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

**Returns:**

The byte at the buffer's current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if the buffer's current position is not smaller than its limit.

Implements **decaf::nio::ByteBuffer** (p. 1041).

**6.166.3.14** `virtual char decaf::internal::nio::ByteBuffer::getChar (int index)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[inline, virtual]`

Reads one byte at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the byte is to be read.

**Returns:**

the char at the given index in the buffer

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1042).

**6.166.3.15** virtual char decaf::internal::nio::ByteArrayBuffer::getChar () throw ( decaf::nio::BufferUnderflowException ) [inline, virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

**Returns:**

the next char in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1042).

**6.166.3.16** virtual double decaf::internal::nio::ByteArrayBuffer::getDouble (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads eight bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the double at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1043).

**6.166.3.17** virtual double decaf::internal::nio::ByteArrayBuffer::getDouble () throw ( decaf::nio::BufferUnderflowException ) [virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next double in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1043).



**6.166.3.18** virtual float decaf::internal::nio::ByteBuffer::getFloat (int *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Reads four bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the float at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1043).

**6.166.3.19** virtual float decaf::internal::nio::ByteBuffer::getFloat () throw (  
decaf::nio::BufferUnderflowException ) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next float in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1044).

**6.166.3.20** virtual int decaf::internal::nio::ByteBuffer::getInt (int *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Reads four bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the int at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1044).

**6.166.3.21** `virtual int decaf::internal::nio::ByteBuffer::getInt () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next int in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1044).

**6.166.3.22** `virtual long long decaf::internal::nio::ByteBuffer::getLong (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads eight bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the long long at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1045).

**6.166.3.23** `virtual long long decaf::internal::nio::ByteBuffer::getLong () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next long long in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1045).

**6.166.3.24** virtual short decaf::internal::nio::ByteBuffer::getShort (int *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Reads two bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the short at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implements **decaf::nio::ByteBuffer** (p. 1045).

**6.166.3.25** virtual short decaf::internal::nio::ByteBuffer::getShort () throw (  
decaf::nio::BufferUnderflowException ) [virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next short in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 1046).

**6.166.3.26** virtual bool decaf::internal::nio::ByteBuffer::hasArray () const  
[inline, virtual]

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ByteBuffer** (p. 1046).

**6.166.3.27** `virtual bool decaf::internal::nio::ByteBuffer::isReadOnly () const`  
[inline, virtual]

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 1046).

**6.166.3.28** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put`  
(int *index*, unsigned char *value*) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given byte into this buffer at the given index.

**Parameters:**

*index* - position in the **Buffer** (p. 928) to write the data

*value* - the byte to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements `decaf::nio::ByteBuffer` (p. 1046).

**6.166.3.29** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put`  
(unsigned char *value*) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

**Parameters:**

*value* - the byte value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements `decaf::nio::ByteBuffer` (p. 1047).

**6.166.3.30** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (int *index*, char *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes one byte containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements decaf::nio::ByteBuffer (p. 1049).

**6.166.3.31** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (char *value*) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements decaf::nio::ByteBuffer (p. 1049).

**6.166.3.32** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (int *index*, double *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data  
*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1050).

**6.166.3.33** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1050).

**6.166.3.34** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data  
*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p.1050).

**6.166.3.35** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (float *value*) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p.1051).

**6.166.3.36** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int *index*, int *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p.1051).

**6.166.3.37** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 1052).

**6.166.3.38** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements `decaf::nio::ByteBuffer` (p. 1052).

**6.166.3.39** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.



**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p.1052).

**6.166.3.40** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (int *index*, short *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data

*value* The value to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p.1053).

**6.166.3.41** virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (short *value*) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

***BufferOverflowException*** (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

***ReadOnlyBufferException*** (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1053).

**6.166.3.42** `virtual void decaf::internal::nio::ByteBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 991) as Read-Only or not Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.166.3.43** `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice () const [virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **ByteBuffer** (p. 1031) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1054).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

## 6.167 decaf::io::ByteArrayInputStream Class Reference

A **ByteArrayInputStream** (p.1020) contains an **internal** (p.144) buffer that contains bytes that may be read from the stream.

#include <src/main/decaf/io/ByteArrayInputStream.h> Inheritance diagram for decaf::io::ByteArrayInputStream:

### Public Member Functions

- **ByteArrayInputStream** ()  
*Creates an **ByteArrayInputStream** (p.1020) with an empty input buffer, the buffer can be initialized with a call to **setByteArray**.*
- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)  
*Creates the input stream and calls **setBuffer** with the specified buffer object.*
- **ByteArrayInputStream** (const unsigned char \*buffer, int bufferSize, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Create an instance of the **ByteArrayInputStream** (p.1020) with the given buffer as the source of input for all read operations.*
- **ByteArrayInputStream** (const unsigned char \*buffer, int bufferSize, int offset, int length, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Create an instance of the **ByteArrayInputStream** (p.1020) with the given buffer as the source of input for all read operations.*
- virtual ~**ByteArrayInputStream** ()
- virtual void **setByteArray** (const std::vector< unsigned char > &buffer)  
*Sets the **internal** (p.144) buffer.*
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.*
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.*
- virtual int **available** () const throw ( IOException )  
*Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.*

**Returns:**

*the number of bytes available on this input stream.*

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

- virtual long long **skip** (long long num) throw ( io::IOException, lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters:**

**num** *The number of bytes to skip.*

**Returns:**

*total bytes skipped*

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters:**

**readLimit** *The max bytes read before marked position is invalid.*

- virtual void **reset** () throw ( IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2142).*

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns:**

*true if this stream instance supports marks*

## Protected Member Functions

- virtual int **doReadByte** () throw ( IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.167.1 Detailed Description

A **ByteArrayInputStream** (p.1020) contains an **internal** (p.144) buffer that contains bytes that may be read from the stream. An **internal** (p.144) counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p.1020) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p.1020) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p.1020) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p.1020) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p.2142).

**Since:**

1.0

### 6.167.2 Constructor & Destructor Documentation

#### 6.167.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ()

Creates an **ByteArrayInputStream** (p.1020) with an empty input buffer, the buffer can be initialized with a call to **setByteArray**.

#### 6.167.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & *buffer*)

Creates the input stream and calls **setBuffer** with the specified buffer object.

**Parameters:**

*buffer* The buffer to be used.

**6.167.2.3** `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, bool own = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`

Create an instance of the **ByteArrayInputStream** (p. 1020) with the given buffer as the source of input for all read operations.

**Parameters:**

*buffer* The initial byte array to use to read from.

*bufferSize* The size of the buffer.

*own* Indicates if this object should take ownership of the array, default is false.

**Exceptions:**

*NullPointerException* if the buffer is Null.

*IllegalArgumentException* if the bufferSize is negative.

**6.167.2.4** `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, int bufferSize, int offset, int length, bool own = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`

Create an instance of the **ByteArrayInputStream** (p. 1020) with the given buffer as the source of input for all read operations.

**Parameters:**

*buffer* The initial byte array to use to read from.

*bufferSize* The size of the buffer.

*offset* The offset into the buffer to begin reading from.

*length* The number of bytes to read past the offset.

*own* Indicates if this object should take ownership of the array, default is false.

**Exceptions:**

*NullPointerException* if the buffer is Null.

*IllegalArgumentException* if the bufferSize is negative.

**6.167.2.5** `virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream () [virtual]`

## 6.167.3 Member Function Documentation

**6.167.3.1** `virtual int decaf::io::ByteArrayInputStream::available () const throw ( IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns:

the number of bytes available on this input stream.

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.167.3.2** `virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2046).

**6.167.3.3** `virtual int decaf::io::ByteArrayInputStream::doReadByte () throw ( IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2046).

**6.167.3.4** `virtual void decaf::io::ByteArrayInputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters:

*readLimit* The max bytes read before marked position is invalid.

Reimplemented from **decaf::io::InputStream** (p. 2046).

**6.167.3.5** `virtual bool decaf::io::ByteArrayInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns:**

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 2047).

**6.167.3.6 virtual void decaf::io::ByteArrayInputStream::reset () throw (IOException) [virtual]**

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2142).

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2049).

**6.167.3.7 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char \* buffer, int bufferSize, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]**

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

**Parameters:**

**buffer** The initial byte array to use to read from.

**bufferSize** The size of the buffer.

**offset** The offset into the buffer to begin reading from.

**length** The number of bytes to read past the offset.

**Exceptions:**

**NullPointerException** if the buffer is Null.



*IllegalArgumentException* if the bufferSize is negative.

**6.167.3.8** virtual void decaf::io::ByteArrayInputStream::setByteArray  
(const unsigned char \* *buffer*, int *bufferSize*)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

**Parameters:**

*buffer* The initial byte array to use to read from.

*bufferSize* The size of the buffer.

**Exceptions:**

*NullPointerException* if the buffer is Null.

*IllegalArgumentException* if the bufferSize is negative.

**6.167.3.9** virtual void decaf::io::ByteArrayInputStream::setByteArray (const  
std::vector< unsigned char > & *buffer*) [virtual]

Sets the **internal** (p. 144) buffer. The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

**Parameters:**

*buffer* The buffer to be used.

**6.167.3.10** virtual long long decaf::io::ByteArrayInputStream::skip  
(long long *num*) throw ( io::IOException,  
lang::exceptions::UnsupportedOperationException )  
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters:**

*num* The number of bytes to skip.

**Returns:**

total bytes skipped

**Exceptions:**

***IOException*** (p. 2142) if an I/O error occurs.

***UnsupportedOperationException*** if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2050).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayInputStream.h`

## 6.168 decaf::io::ByteArrayOutputStream Class Reference

#include <src/main/decaf/io/ByteArrayOutputStream.h> Inheritance diagram for decaf::io::ByteArrayOutputStream:

### Public Member Functions

- **ByteArrayOutputStream** ()  
*Default Constructor - uses a default **internal** (p. 144) buffer of 32 bytes, the size increases as the need for more room arises.*
- **ByteArrayOutputStream** (int bufferSize) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteArrayOutputStream** (p. 1028) with an **internal** (p. 144) buffer allocated with the given size.*
- virtual ~**ByteArrayOutputStream** ()
- std::pair< unsigned char \*, int > **toByteArray** () const  
*Creates a newly allocated byte array.*
- long long **size** () const  
*Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 1028).*
- virtual void **reset** () throw ( IOException )  
*Clear current Stream contents.*
- virtual std::string **toString** () const  
*Converts the bytes in the buffer into a standard C++ string.*
- void **writeTo** (OutputStream \*out) const throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write( buf, 0, count ).*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.168.1 Constructor & Destructor Documentation

#### 6.168.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default **internal** (p. 144) buffer of 32 bytes, the size increases as the need for more room arises.

### 6.168.1.2 `decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (int bufferSize) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a `ByteArrayOutputStream` (p. 1028) with an `internal` (p. 144) buffer allocated with the given size.

#### Parameters:

*bufferSize* The size to use for the `internal` (p. 144) buffer.

#### Exceptions:

*IllegalArgumentException* if the size is less than or equal to zero.

### 6.168.1.3 `virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream () [virtual]`

## 6.168.2 Member Function Documentation

### 6.168.2.1 `virtual void decaf::io::ByteArrayOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]`

Reimplemented from `decaf::io::OutputStream` (p. 2909).

### 6.168.2.2 `virtual void decaf::io::ByteArrayOutputStream::doWriteByte (unsigned char value) throw ( decaf::io::IOException ) [protected, virtual]`

Implements `decaf::io::OutputStream` (p. 2910).

### 6.168.2.3 `virtual void decaf::io::ByteArrayOutputStream::reset () throw ( IOException ) [virtual]`

Clear current Stream contents.

#### Exceptions:

*IOException* (p. 2142)

### 6.168.2.4 `long long decaf::io::ByteArrayOutputStream::size () const`

Gets the current count of bytes written into this `ByteArrayOutputStream` (p. 1028).

#### Returns:

the number of valid bytes contained in the `ByteArrayOutputStream` (p. 1028).

**6.168.2.5** `std::pair<unsigned char*, int> decaf::io::ByteArrayOutputStream::toByteArray () const`

Creates a newly allocated byte array. Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

**Returns:**

an STL pair containing the copied array and its size.

**6.168.2.6** `virtual std::string decaf::io::ByteArrayOutputStream::toString () const [virtual]`

Converts the bytes in the buffer into a standard C++ string.

**Returns:**

a string containing the bytes in the buffer

Reimplemented from `decaf::io::OutputStream` (p. 2911).

**6.168.2.7** `void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * out) const throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write( buf, 0, count )`.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayOutputStream.h`

## 6.169 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

```
#include <src/main/decaf/nio/ByteBuffer.h> Inheritance diagram for decaf::nio::ByteBuffer:
```

### Public Member Functions

- virtual **~ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer** & **get** (std::vector< unsigned char > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **ByteBuffer** & **get** (unsigned char \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- **ByteBuffer** & **put** (**ByteBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the bytes remaining in the given source buffer into this buffer.*
- **ByteBuffer** & **put** (const unsigned char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers bytes into this buffer from the given source array.*
- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source byte array into this buffer.*
- virtual bool **isReadOnly** () const =0  
*Tells whether or not this buffer is read-only.*
- virtual unsigned char \* **array** ()=0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the byte array that backs this buffer.*
- virtual int **arrayOffset** () const =0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the offset within this buffer's backing array of the first element of the buffer.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible byte array.*
- virtual **CharBuffer** \* **asCharBuffer** () const =0

*Creates a view of this byte buffer as a char buffer.*

- virtual **DoubleBuffer** \* **asDoubleBuffer** () const =0  
*Creates a view of this byte buffer as a double buffer.*
- virtual **FloatBuffer** \* **asFloatBuffer** () const =0  
*Creates a view of this byte buffer as a float buffer.*
- virtual **IntBuffer** \* **asIntBuffer** () const =0  
*Creates a view of this byte buffer as a int buffer.*
- virtual **LongBuffer** \* **asLongBuffer** () const =0  
*Creates a view of this byte buffer as a long buffer.*
- virtual **ShortBuffer** \* **asShortBuffer** () const =0  
*Creates a view of this byte buffer as a short buffer.*
- virtual **ByteBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only byte buffer that shares this buffer's content.*
- virtual **ByteBuffer** & **compact** ()=0 throw ( **ReadOnlyBufferException** )  
*Compacts this buffer.*
- virtual **ByteBuffer** \* **duplicate** ()=0  
*Creates a new byte buffer that shares this buffer's content.*
- virtual unsigned char **get** () const =0 throw ( **BufferUnderflowException** )  
*Relative get method.*
- virtual unsigned char **get** (int index) const =0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** )  
*Absolute get method.*
- virtual char **getChar** ()=0 throw ( **BufferUnderflowException** )  
*Reads the next byte at this buffer's current position, and then increments the position by one.*
- virtual char **getChar** (int index) const =0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** )  
*Reads one byte at the given index and returns it.*
- virtual double **getDouble** ()=0 throw ( **BufferUnderflowException** )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*
- virtual double **getDouble** (int index) const =0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** )  
*Reads eight bytes at the given index and returns it.*
- virtual float **getFloat** ()=0 throw ( **BufferUnderflowException** )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

- virtual float **getFloat** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual long long **getLong** ()=0 throw ( BufferUnderflowException )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*
- virtual long long **getLong** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual int **getInt** ()=0 throw ( BufferUnderflowException )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*
- virtual int **getInt** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual short **getShort** ()=0 throw ( BufferUnderflowException )  
*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*
- virtual short **getShort** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given index and returns it.*
- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the current position, and then increments the position.*
- virtual **ByteBuffer** & **put** (int index, unsigned char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the given index.*
- virtual **ByteBuffer** & **putChar** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*
- virtual **ByteBuffer** & **putChar** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes one byte containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** & **putDouble** (double value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*



- virtual **ByteBuffer** & **putDouble** (int index, double value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** & **putFloat** (float value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*
- virtual **ByteBuffer** & **putFloat** (int index, float value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** & **putLong** (long long value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*
- virtual **ByteBuffer** & **putLong** (int index, long long value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** & **putInt** (int value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*
- virtual **ByteBuffer** & **putInt** (int index, int value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** & **putShort** (short value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*
- virtual **ByteBuffer** & **putShort** (int index, short value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes two bytes containing the given value, into this buffer at the given index.*
- virtual **ByteBuffer** \* **slice** () const =0  
*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **ByteBuffer** &value) const
- virtual bool **equals** (const **ByteBuffer** &value) const
- virtual bool **operator==** (const **ByteBuffer** &value) const

- virtual bool **operator**< (const **ByteBuffer** &value) const

## Static Public Member Functions

- static **ByteBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **ByteBuffer** \* **wrap** (unsigned char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **ByteBuffer** (p. 1031).*
- static **ByteBuffer** \* **wrap** (std::vector< unsigned char > &buffer)  
*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 1031).*

## Protected Member Functions

- **ByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteBuffer** (p. 1031) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.169.1 Detailed Description

This class defines six categories of operations upon byte buffers:. 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat**() (p. 1044) float **getFloat**(int index) void **putFloat**(float f) (p. 1051) void **putFloat**(int index, float f) (p. 1050)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the **FloatBuffer** (p.1925) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

## 6.169.2 Constructor & Destructor Documentation

### 6.169.2.1 decaf::nio::ByteBuffer::ByteBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **ByteBuffer** (p.1031) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

#### Parameters:

*capacity* The size of the array, this is the limit we read and write to.

#### Exceptions:

*IllegalArgumentException* if capacity is negative.

### 6.169.2.2 virtual decaf::nio::ByteBuffer::~~ByteBuffer () [inline, virtual]

## 6.169.3 Member Function Documentation

### 6.169.3.1 static ByteBuffer\* decaf::nio::ByteBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters:

*capacity* The **internal** (p.144) buffer's capacity.

#### Returns:

a newly allocated **ByteBuffer** (p.1031) which the caller owns.

**Exceptions:**

*IllegalArgumentException* if capacity is negative.

**6.169.3.2** `virtual unsigned char* decaf::nio::ByteBuffer::array  
() throw ( ReadOnlyBufferException, de-  
caf::lang::exceptions::UnsupportedOperationException )  
[pure virtual]`

Returns the byte array that backs this buffer. Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The array that backs this buffer

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is backed by an array but is read-only  
*UnsupportedOperationException* if this buffer is not backed by an accessible array

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1003).

**6.169.3.3** `virtual int decaf::nio::ByteBuffer::arrayOffset  
() const throw ( ReadOnlyBufferException,  
decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 1037).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset within this buffer's array of the first element of the buffer.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is backed by an array but is read-only  
*UnsupportedOperationException* if this buffer is not backed by an accessible array.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1004).

**6.169.3.4** `virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const [pure  
virtual]`

Creates a view of this byte buffer as a char buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new Char **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1004).

**6.169.3.5 virtual DoubleBuffer\* decaf::nio::ByteBuffer::asDoubleBuffer () const**  
[pure virtual]

Creates a view of this byte buffer as a double buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new double **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1004).

**6.169.3.6 virtual FloatBuffer\* decaf::nio::ByteBuffer::asFloatBuffer () const** [pure virtual]

Creates a view of this byte buffer as a float buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new float **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1005).

**6.169.3.7 virtual IntBuffer\* decaf::nio::ByteBuffer::asIntBuffer () const** [pure virtual]

Creates a view of this byte buffer as a int buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new int **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1005).

**6.169.3.8 virtual LongBuffer\* decaf::nio::ByteBuffer::asLongBuffer () const [pure virtual]**

Creates a view of this byte buffer as a long buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new long **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1006).

**6.169.3.9 virtual ByteBuffer\* decaf::nio::ByteBuffer::asReadOnlyBuffer () const [pure virtual]**

Creates a new, read-only byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1006).

**6.169.3.10 virtual ShortBuffer\* decaf::nio::ByteBuffer::asShortBuffer () const [pure virtual]**

Creates a view of this byte buffer as a short buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the new short **Buffer** (p. 928), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1006).

### 6.169.3.11 virtual ByteBuffer& decaf::nio::ByteBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns:

a reference to this **ByteBuffer** (p. 1031).

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1007).

### 6.169.3.12 virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer & value) const [virtual]

### 6.169.3.13 virtual ByteBuffer\* decaf::nio::ByteBuffer::duplicate () [pure virtual]

Creates a new byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

a new Byte **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1007).

### 6.169.3.14 virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & value) const [virtual]

### 6.169.3.15 virtual unsigned char decaf::nio::ByteBuffer::get (int index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the byte at the given index.

#### Parameters:

*index* The index in the **Buffer** (p. 928) where the byte is to be read.

**Returns:**

the byte that is located at the given index.

**Exceptions:**

***IndexOutOfBoundsException*** if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1007).

### 6.169.3.16 **virtual unsigned char decaf::nio::ByteBuffer::get () const throw ( BufferUnderflowException )** [pure virtual]

Relative get method. Reads the byte at this buffer's current position, and then increments the position.

**Returns:**

The byte at the buffer's current position.

**Exceptions:**

***BufferUnderflowException*** (p. 957) if the buffer's current position is not smaller than its limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1008).

### 6.169.3.17 **ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method. This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

**Parameters:**

***buffer*** The pointer to an allocated buffer to fill.  
***size*** The size of the passed in **Buffer** (p. 928).  
***offset*** The position in the buffer to start filling.  
***length*** The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

***IndexOutOfBoundsException*** if the preconditions of `size`, `offset`, or `length` are not met.



*BufferUnderflowException* (p. 957) if there are fewer than length bytes remaining in this buffer.

*NullPointerException* if the passed buffer is null.

#### 6.169.3.18 `ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > buffer) throw ( BufferUnderflowException )`

Relative bulk get method. This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

##### Returns:

a reference to this Byte **Buffer** (p. 928).

##### Exceptions:

*BufferUnderflowException* (p. 957) if there are fewer than length bytes remaining in this buffer

#### 6.169.3.19 `virtual char decaf::nio::ByteBuffer::getChar (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads one byte at the given index and returns it.

##### Parameters:

*index* The index in the **Buffer** (p. 928) where the byte is to be read.

##### Returns:

the char at the given index in the buffer

##### Exceptions:

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1008).

#### 6.169.3.20 `virtual char decaf::nio::ByteBuffer::getChar () throw ( BufferUnderflowException ) [pure virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

##### Returns:

the next char in the buffer.

##### Exceptions:

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1009).

**6.169.3.21** `virtual double decaf::nio::ByteBuffer::getDouble (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads eight bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the double at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1009).

**6.169.3.22** `virtual double decaf::nio::ByteBuffer::getDouble () throw ( BufferUnderflowException ) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next double in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1009).

**6.169.3.23** `virtual float decaf::nio::ByteBuffer::getFloat (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads four bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the float at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1010).

**6.169.3.24** virtual float decaf::nio::ByteBuffer::getFloat () throw (   
BufferUnderflowException ) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next float in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1010).

**6.169.3.25** virtual int decaf::nio::ByteBuffer::getInt (int *index*) const throw (   
decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Reads four bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the int at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1010).

**6.169.3.26** virtual int decaf::nio::ByteBuffer::getInt () throw (   
BufferUnderflowException ) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next int in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1011).

**6.169.3.27** `virtual long long decaf::nio::ByteBuffer::getLong (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads eight bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the long long at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1011).

**6.169.3.28** `virtual long long decaf::nio::ByteBuffer::getLong () throw ( BufferUnderflowException ) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next long long in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1011).

**6.169.3.29** `virtual short decaf::nio::ByteBuffer::getShort (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads two bytes at the given index and returns it.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the bytes are to be read.

**Returns:**

the short at the given index in the buffer.

**Exceptions:**

*IndexOutOfBoundsException* if there are not enough bytes remaining to fill the requested Data Type, or index is negative.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1012).

**6.169.3.30** virtual short decaf::nio::ByteBuffer::getShort () throw ( BufferUnderflowException ) [pure virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

**Returns:**

the next short in the buffer.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1012).

**6.169.3.31** virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible byte array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1012).

**6.169.3.32** virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 931).

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1013).

**6.169.3.33** virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & value) const [virtual]**6.169.3.34** virtual bool decaf::nio::ByteBuffer::operator== (const ByteBuffer & value) const [virtual]**6.169.3.35** virtual ByteBuffer& decaf::nio::ByteBuffer::put (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes the given byte into this buffer at the given index.

**Parameters:**

*index* - position in the **Buffer** (p. 928) to write the data

*value* - the byte to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1013).

**6.169.3.36** `virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char value)  
throw ( BufferOverflowException, ReadOnlyBufferException )` [pure  
virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

**Parameters:**

*value* - the byte value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1013).

**6.169.3.37** `ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char >  
& buffer) throw ( BufferOverflowException, ReadOnlyBufferException )`

This method transfers the entire content of the given source byte array into this buffer. This is the same as calling `put( &buffer[0], buffer.size(), 0, buffer.size() )`

**Parameters:**

*buffer* The buffer whose contents are copied to this **ByteBuffer** (p. 1031).

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

**6.169.3.38** `ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

This method transfers bytes into this buffer from the given source array. If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no bytes are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters:**

*buffer* The array from which bytes are to be read.

*size* The size of the given array.

*offset* The offset within the array of the first byte to be read.

*length* The number of bytes to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**NullPointerException** if the passed buffer is null.

**IndexOutOfBoundsException** if the preconditions of `size`, `offset`, or `length` are not met.

**6.169.3.39** `ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the bytes remaining in the given source buffer into this buffer. If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 933), then no bytes are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

**Parameters:**

*src* The buffer to take bytes from an place in this one.

**Returns:**

a reference to this buffer

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer for the remaining bytes in the source buffer

*IllegalArgumentException* if the source buffer is this buffer

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

**6.169.3.40** `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1014).

**6.169.3.41** `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1014).



**6.169.3.42** virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (int *index*, double *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1014).

**6.169.3.43** virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double *value*) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1015).

**6.169.3.44** virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (int *index*, float *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1015).

**6.169.3.45** `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1016).

**6.169.3.46** `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1016).

**6.169.3.47** `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1017).

**6.169.3.48** `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The value to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 1017).

**6.169.3.49** `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1017).

**6.169.3.50** `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data

*value* The value to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1018).

**6.169.3.51** `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters:**

*value* The value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there are fewer than bytes remaining in this buffer than the size of the data to be written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1018).

**6.169.3.52 virtual ByteBuffer\* decaf::nio::ByteBuffer::slice () const [pure virtual]**

Creates a new byte buffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **ByteBuffer** (p.1031) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p.1019).

**6.169.3.53 virtual std::string decaf::nio::ByteBuffer::toString () const [virtual]****Returns:**

a std::string describing this object

**6.169.3.54 static ByteBuffer\* decaf::nio::ByteBuffer::wrap (std::vector< unsigned char > & *buffer*) [static]**

Wraps the passed STL Byte Vector in a **ByteBuffer** (p.1031). The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new **ByteBuffer** (p.1031) that is backed by *buffer*, caller owns.

**6.169.3.55 static ByteBuffer\* decaf::nio::ByteBuffer::wrap (unsigned char \* *array*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]**

Wraps the passed buffer with a new **ByteBuffer** (p.1031). The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*array* The array that will back the new buffer.

*size* The size of the provided array.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new **ByteBuffer** (p. 1031) that is backed by buffer, caller owns.

**Exceptions:**

*NullPointerException* if the array passed in is NULL.

*IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ByteBuffer.h**

## 6.170 cms::BytesMessage Class Reference

A **BytesMessage** (p.1056) object is used to send a message containing a stream of unsigned bytes.

#include <src/main/cms/BytesMessage.h> Inheritance diagram for cms::BytesMessage:

### Public Member Functions

- virtual **~BytesMessage** ()
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.*
- virtual int **getBodyLength** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Returns the number of bytes contained in the body of this message.*
- virtual void **reset** ()=0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Bytes message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the bytes message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Bytes message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the bytes message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the bytes message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*

- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a portion of the bytes message stream.*

- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a portion of a byte array to the bytes message stream.*

- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Char from the Bytes message stream.*

- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a char to the bytes message stream as a 1-byte value.*

- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit float from the Bytes message stream.*

- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a float to the bytes message stream as a 4 byte value.*

- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Bytes message stream.*

- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the bytes message stream as a 8 byte value.*

- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Bytes message stream.*

- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the bytes message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Bytes message stream.*

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the bytes message stream as a 2 byte value.*



- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit signed integer from the Bytes message stream.*
- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed int to the bytes message stream as a 4 byte value.*
- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Bytes message stream.*
- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a long long to the bytes message stream as a 8 byte value.*
- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an ASCII String from the Bytes message stream.*
- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an ASCII String to the Bytes message stream.*
- virtual std::string **readUTF** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an UTF String from the **BytesMessage** (p. 1056) stream.*
- virtual void **writeUTF** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an UTF String to the **BytesMessage** (p. 1056) stream.*
- virtual **BytesMessage** \* **clone** () const =0  
*Clones this message.*

### 6.170.1 Detailed Description

A **BytesMessage** (p.1056) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p.2534) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 1056) interface.

The **BytesMessage** (p.1056) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1566) and **decaf.io.DataOutputStream** (p. 1579).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when `clearBody` is called, the body of the message is in write-only mode. After the first call to `reset` has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called `reset` so that the message body is in read-only mode for the client.

If `clearBody` is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2723) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2724) is thrown.

**Since:**

1.0

## 6.170.2 Constructor & Destructor Documentation

**6.170.2.1** `virtual cms::BytesMessage::~~BytesMessage () [inline, virtual]`

## 6.170.3 Member Function Documentation

**6.170.3.1** `virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]`

Clones this message.

**Returns:**

a deep copy of this message.

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs while cloning the **Message** (p. 2534).

Implements **cms::Message** (p. 2539).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 236).

**6.170.3.2** `virtual unsigned char* cms::BytesMessage::getBodyBytes () const throw ( cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller. This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

**Returns:**

pointer to a byte buffer that the call owns upon completion of this method.

**Exceptions:**

*CMSException* (p. 1160) - If an internal error occurs.

*MessageNotReadableException* (p. 2723) - If the message is in Write Only Mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 237).

**6.170.3.3** `virtual int cms::BytesMessage::getBodyLength () const throw ( cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Returns the number of bytes contained in the body of this message.

**Returns:**

number of bytes.

**Exceptions:**

*CMSException* (p. 1160) - If an internal error occurs.

*MessageNotReadableException* (p. 2723) - If the message is in Write Only Mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 238).

**6.170.3.4** `virtual bool cms::BytesMessage::readBoolean () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Boolean from the Bytes message stream.

**Returns:**

boolean value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 238).

**6.170.3.5** `virtual unsigned char cms::BytesMessage::readByte () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Byte from the Bytes message stream.

**Returns:**

unsigned char value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 239).

**6.170.3.6** `virtual int cms::BytesMessage::readBytes (unsigned char *  
buffer, int length) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException ) [pure  
virtual]`

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

#### Parameters:

*buffer* the buffer into which the data is read

*length* the number of bytes to read; must be less than or equal to value.length

#### Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions:

*CMSEException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 239).

**6.170.3.7** `virtual int cms::BytesMessage::readBytes (std::vector< unsigned  
char > & value) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException ) [pure  
virtual]`

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

**Parameters:**

*value* buffer to place data in

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSEException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 240).

**6.170.3.8** `virtual char cms::BytesMessage::readChar () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a Char from the Bytes message stream.

**Returns:**

char value from stream

**Exceptions:**

*CMSEException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 240).

**6.170.3.9** `virtual double cms::BytesMessage::readDouble () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 64 bit double from the Bytes message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSEException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 240).

**6.170.3.10** `virtual float cms::BytesMessage::readFloat () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 32 bit float from the Bytes message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 241).

**6.170.3.11** `virtual int cms::BytesMessage::readInt () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

**Returns:**

int value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 241).

**6.170.3.12** `virtual long long cms::BytesMessage::readLong () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 64 bit long from the Bytes message stream.

**Returns:**

long long value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 241).

**6.170.3.13** virtual short cms::BytesMessage::readShort () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

**Returns:**

short value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 242).

**6.170.3.14** virtual std::string cms::BytesMessage::readString () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads an ASCII String from the Bytes message stream.

**Returns:**

String from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 242).

**6.170.3.15** virtual unsigned short cms::BytesMessage::readUnsignedShort () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

**Returns:**

unsigned short value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 243).

**6.170.3.16** `virtual std::string cms::BytesMessage::readUTF () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads an UTF String from the `BytesMessage` (p. 1056) stream.

**Returns:**

String from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 243).

**6.170.3.17** `virtual void cms::BytesMessage::reset () throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

**Exceptions:**

*CMSException* (p. 1160) - If the provider fails to perform the reset operation.

*MessageFormatException* (p. 2662) - If the `Message` (p. 2534) has an invalid format.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 243).

**6.170.3.18** `virtual void cms::BytesMessage::setBodyBytes (const unsigned char * buffer, int numBytes) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

sets the bytes given to the message body.

**Parameters:**

*buffer* Byte Buffer to copy

*numBytes* Number of bytes in Buffer to copy

**Exceptions:**

*CMSException* (p. 1160) - If an internal error occurs.

*MessageNotWriteableException* (p. 2724) - if in Read Only Mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 244).



**6.170.3.19** `virtual void cms::BytesMessage::writeBoolean (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters:**

*value* boolean to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 244).

**6.170.3.20** `virtual void cms::BytesMessage::writeByte (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

**Parameters:**

*value* byte to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 244).

**6.170.3.21** `virtual void cms::BytesMessage::writeBytes (const unsigned char * value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

*offset* the initial offset within the byte array

*length* the number of bytes to use

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 245).

**6.170.3.22** `virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 245).

**6.170.3.23** `virtual void cms::BytesMessage::writeChar (char value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a char to the bytes message stream as a 1-byte value.

**Parameters:**

*value* char to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 245).

**6.170.3.24** `virtual void cms::BytesMessage::writeDouble (double value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a double to the bytes message stream as a 8 byte value.

**Parameters:**

*value* double to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 246).

**6.170.3.25** `virtual void cms::BytesMessage::writeFloat (float value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a float to the bytes message stream as a 4 byte value.

**Parameters:**

*value* float to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 246).

**6.170.3.26** `virtual void cms::BytesMessage::writeInt (int value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

**Parameters:**

*value* signed int to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 246).

**6.170.3.27** `virtual void cms::BytesMessage::writeLong (long long value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

**Parameters:**

*value* signed long long to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 247).

**6.170.3.28** `virtual void cms::BytesMessage::writeShort (short value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

**Parameters:**

*value* signed short to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 247).

**6.170.3.29** `virtual void cms::BytesMessage::writeString (const std::string & value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes an ASCII String to the Bytes message stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 247).

**6.170.3.30** `virtual void cms::BytesMessage::writeUnsignedShort (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

**Parameters:**

*value* unsigned short to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 248).

**6.170.3.31**    `virtual void cms::BytesMessage::writeUTF (const std::string & value)  
                 throw ( cms::MessageNotWriteableException, cms::CMSException )  
                 [pure virtual]`

Writes an UTF String to the **BytesMessage** (p. 1056) stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of bytes stream has been reached.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 248).

The documentation for this class was generated from the following file:

- `src/main/cms/BytesMessage.h`

## 6.171 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

#include <src/main/activemq/cmsutil/CachedConsumer.h> Inheritance diagram for activemq::cmsutil::CachedConsumer:

### Public Member Functions

- **CachedConsumer** (**cms::MessageConsumer** \*consumer)
- virtual **~CachedConsumer** ()
- virtual void **close** () throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual **cms::Message** \* **receive** () throw ( cms::CMSEException )  
*Synchronously Receive a Message.*
- virtual **cms::Message** \* **receive** (int millisecs) throw ( cms::CMSEException )  
*Synchronously Receive a Message, time out after defined interval.*
- virtual **cms::Message** \* **receiveNoWait** () throw ( cms::CMSEException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**cms::MessageListener** \*listener) throw ( cms::CMSEException )  
*Sets the MessageListener that this class will send notifs on.*
- virtual **cms::MessageListener** \* **getMessageListener** () const throw ( cms::CMSEException )  
*Gets the MessageListener that this class will send new Message notification events to.*
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*

### Protected Member Functions

- **CachedConsumer** (const **CachedConsumer** &)
- **CachedConsumer** & **operator=** (const **CachedConsumer** &)

#### 6.171.1 Detailed Description

A cached message consumer contained within a pooled session.

## 6.171.2 Constructor & Destructor Documentation

- 6.171.2.1** `activemq::cmsutil::CachedConsumer::CachedConsumer (const CachedConsumer &) [inline, protected]`
- 6.171.2.2** `activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer * consumer) [inline]`
- 6.171.2.3** `virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer () [inline, virtual]`

## 6.171.3 Member Function Documentation

- 6.171.3.1** `virtual void activemq::cmsutil::CachedConsumer::close () throw (cms::CMSEException) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 1148).

- 6.171.3.2** `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener () const throw (cms::CMSEException) [inline, virtual]`

Gets the MessageListener that this class will send new Message notification events to.

### Returns:

The listener of messages received by this consumer

### Exceptions:

*CMSEException* - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2590).

- 6.171.3.3** `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector () const throw (cms::CMSEException) [inline, virtual]`

Gets this message consumer's message selector expression.

### Returns:

This Consumer's selector expression or "".

### Exceptions:

*CMSEException* - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2590).

**6.171.3.4** `CachedConsumer& activemq::cmsutil::CachedConsumer::operator=`  
`(const CachedConsumer &) [inline, protected]`

**6.171.3.5** `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int`  
`milliseconds) throw ( cms::CMSEException ) [inline, virtual]`

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2590).

**6.171.3.6** `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive ()`  
`throw ( cms::CMSEException ) [inline, virtual]`

Synchronously Receive a Message.

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2591).

**6.171.3.7** `virtual cms::Message* ac-`  
`tivemq::cmsutil::CachedConsumer::receiveNoWait ()`  
`throw ( cms::CMSEException ) [inline, virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2591).

**6.171.3.8** `virtual void activemq::cmsutil::CachedConsumer::setMessageListener`  
`(cms::MessageListener * listener) throw ( cms::CMSEException )`  
`[inline, virtual]`

Sets the MessageListener that this class will send notifs on.



**Parameters:**

*listener* The listener of messages received by this consumer.

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2591).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedConsumer.h**

## 6.172 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

#include <src/main/activemq/cmsutil/CachedProducer.h> Inheritance diagram for activemq::cmsutil::CachedProducer:

### Public Member Functions

- **CachedProducer** (**cms::MessageProducer** \*producer)
- virtual **~CachedProducer** ()
- virtual void **close** () throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual void **send** (**cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSEException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSEException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSEException )  
*Sets if Message Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const throw ( cms::CMSEException )  
*Gets if Message Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )

*Gets if Message Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*

## Protected Member Functions

- **CachedProducer** (const **CachedProducer** &)
- **CachedProducer** & **operator=** (const **CachedProducer** &)

### 6.172.1 Detailed Description

A cached message producer contained within a pooled session.

### 6.172.2 Constructor & Destructor Documentation

- 6.172.2.1 **activemq::cmsutil::CachedProducer::CachedProducer** (const **CachedProducer** &) [inline, protected]
- 6.172.2.2 **activemq::cmsutil::CachedProducer::CachedProducer** (cms::MessageProducer \* *producer*) [inline]
- 6.172.2.3 **virtual activemq::cmsutil::CachedProducer::~~CachedProducer** () [inline, virtual]

### 6.172.3 Member Function Documentation

- 6.172.3.1 **virtual void activemq::cmsutil::CachedProducer::close** () throw ( cms::CMSEException ) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p.1148).

- 6.172.3.2 **virtual int activemq::cmsutil::CachedProducer::getDeliveryMode** () const throw ( cms::CMSEException ) [inline, virtual]

Gets the delivery mode for this Producer.

**Returns:**

The DeliveryMode

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2726).

**6.172.3.3** `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID () const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

**Returns:**

boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2727).

**6.172.3.4** `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp () const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

**Returns:**

boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2727).

**6.172.3.5** `virtual int activemq::cmsutil::CachedProducer::getPriority () const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

**Returns:**

int based priority level

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2727).

**6.172.3.6** `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive ()  
const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

**Returns:**

Time to live value in milliseconds

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2728).

**6.172.3.7** `CachedProducer& activemq::cmsutil::CachedProducer::operator= (const  
CachedProducer &) [inline, protected]`

**6.172.3.8** `virtual void activemq::cmsutil::CachedProducer::send (const  
cms::Destination * destination, cms::Message * message, int  
deliveryMode, int priority, long long timeToLive) throw (  
cms::CMSEException ) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*destination* The destination on which to send the message

*message* The message to be sent.

*deliveryMode* The delivery mode to be used.

*priority* The priority for this message.

*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

*CMSEException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2728).

**6.172.3.9** `virtual void activemq::cmsutil::CachedProducer::send (const  
cms::Destination * destination, cms::Message * message) throw (  
cms::CMSEException ) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

**Parameters:**

*destination* The destination on which to send the message

*message* the message to be sent.

**Exceptions:**

***CMSEException*** - if an internal error occurs while sending the message.

***MessageFormatException*** - if an Invalid Message is given.

***InvalidDestinationException*** - if a client uses this method with a MessageProducer with an invalid destination.

***UnsupportedOperationException*** - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2728).

**6.172.3.10** `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )` [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*message* The message to be sent.

*deliveryMode* The delivery mode to be used.

*priority* The priority for this message.

*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

***CMSEException*** - if an internal error occurs while sending the message.

***MessageFormatException*** - if an Invalid Message is given.

***InvalidDestinationException*** - if a client uses this method with a MessageProducer with an invalid destination.

***UnsupportedOperationException*** - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2729).

**6.172.3.11** `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message) throw ( cms::CMSEException )` [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

**Parameters:**

*message* The message to be sent.

**Exceptions:**

*CMSEException* - if an internal error occurs while sending the message.

*MessageFormatException* - if an Invalid Message is given.

*InvalidDestinationException* - if a client uses this method with a MessageProducer with an invalid destination.

*UnsupportedOperationException* - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2729).

**6.172.3.12** `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int mode) throw ( cms::CMSEException ) [inline, virtual]`

Sets the delivery mode for this Producer.

**Parameters:**

*mode* The DeliveryMode

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2730).

**6.172.3.13** `virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool value) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

**Parameters:**

*value* boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2730).

**6.172.3.14** `virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool value) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

**Parameters:**

*value* - boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2730).

**6.172.3.15** `virtual void activemq::cmsutil::CachedProducer::setPriority (int priority) throw ( cms::CMSEException )` [inline, virtual]

Sets the Priority that this Producers sends messages at.

**Parameters:**

*priority* int value for Priority level

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2731).

**6.172.3.16** `virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long time) throw ( cms::CMSEException )` [inline, virtual]

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

**Parameters:**

*time* default time to live value in milliseconds

**Exceptions:**

*CMSEException* - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2731).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedProducer.h`



## 6.173 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

### Public Member Functions

- virtual `~Callable ()`
- virtual `V call ()=0 throw ( decaf::lang::Exception )`  
*Computes a result, or throws an exception if unable to do so.*

### 6.173.1 Detailed Description

```
template<typename V> class decaf::util::concurrent::Callable< V >
```

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called `call`. This interface differs from the `Runnable` interface in that a **Callable** (p.1082) object can return a result and is allowed to throw an exceptions from its `call` method.

The `Executors` class contains utility methods to convert from other common forms to **Callable** (p.1082) classes.

Since:

1.0

### 6.173.2 Constructor & Destructor Documentation

**6.173.2.1** `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable () [inline, virtual]`

### 6.173.3 Member Function Documentation

**6.173.3.1** `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call () throw ( decaf::lang::Exception ) [pure virtual]`

Computes a result, or throws an exception if unable to do so.

Returns:

Computed Result.

Exceptions:

*Exception* If unable to compute a result.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

## 6.174 decaf::util::concurrent::CancellationException Class Reference

#include <src/main/decaf/util/concurrent/CancellationException.h> Inheritance diagram for decaf::util::concurrent::CancellationException:

### Public Member Functions

- **CancellationException** () throw ()  
*Default Constructor.*
- **CancellationException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CancellationException** (const **CancellationException** &ex) throw ()  
*Copy Constructor.*
- **CancellationException** (const std::exception \*cause) throw ()  
*Constructor.*
- **CancellationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **CancellationException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CancellationException** \* clone () const  
*Clones this exception.*
- virtual ~**CancellationException** () throw ()

### 6.174.1 Constructor & Destructor Documentation

#### 6.174.1.1 decaf::util::concurrent::CancellationException::CancellationException () throw () [inline]

Default Constructor.

#### 6.174.1.2 decaf::util::concurrent::CancellationException::CancellationException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

#### 6.174.1.3 `decaf::util::concurrent::CancellationException::CancellationException` (const CancellationException & *ex*) throw () [inline]

Copy Constructor.

##### Parameters:

*ex* - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

#### 6.174.1.4 `decaf::util::concurrent::CancellationException::CancellationException` (const std::exception \* *cause*) throw () [inline]

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.174.1.5 `decaf::util::concurrent::CancellationException::CancellationException` (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*msg* - The message to report

... - list of primitives that are formatted into the message

#### 6.174.1.6 `decaf::util::concurrent::CancellationException::CancellationException` (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*cause* - The exception that was the cause for this one to be thrown.

*msg* - The message to report

... - list of primitives that are formatted into the message

**6.174.1.7**    **virtual**  
          **decaf::util::concurrent::CancellationException::~~CancellationException ()**  
          **throw ()**    [inline, virtual]

## **6.174.2    Member Function Documentation**

**6.174.2.1**    **virtual CancellationException\* de-**  
          **caf::util::concurrent::CancellationException::clone () const**  
          [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

        a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CancellationException.h`

## 6.175 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

#include <src/main/decaf/security/cert/Certificate.h>Inheritance diagram for decaf::security::cert::Certificate:

### Public Member Functions

- virtual **~Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0  
*Compares the encoded form of the two certificates.*
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw ( CertificateEncodingException )  
*Provides the encoded form of this certificate.*
- virtual std::string **getType** () const =0  
*Returns the type of this certificate.*
- virtual **PublicKey** \* **getPublicKey** ()=0  
*Gets the public key of this certificate.*
- virtual const **PublicKey** \* **getPublicKey** () const =0  
*Gets the public key of this certificate.*
- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException )  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException )  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual std::string **toString** () const =0  
*Returns a string representation of this certificate.*

### 6.175.1 Detailed Description

Base interface for all identity certificates.

## 6.175.2 Constructor & Destructor Documentation

**6.175.2.1** `virtual decaf::security::cert::Certificate::~~Certificate () [inline, virtual]`

## 6.175.3 Member Function Documentation

**6.175.3.1** `virtual bool decaf::security::cert::Certificate::equals (const Certificate & cert) const [pure virtual]`

Compares the encoded form of the two certificates.

### Parameters:

*cert* (p. 167) The certificate to be tested for equality with this certificate.

### Returns:

true if the given certificate is equal to this certificate.

**6.175.3.2** `virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & output) const throw ( CertificateEncodingException ) [pure virtual]`

Provides the encoded form of this certificate.

### Parameters:

*output* Receives the encoded form of this certificate.

### Exceptions:

*CertificateEncodingException* (p. 1090) if an encoding error occurs

**6.175.3.3** `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const [pure virtual]`

Gets the public key of this certificate.

### Returns:

the public key

**6.175.3.4** `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]`

Gets the public key of this certificate.

### Returns:

the public key

**6.175.3.5** `virtual std::string decaf::security::cert::Certificate::getType () const`  
[pure virtual]

Returns the type of this certificate.

**Returns:**

the type of this certificate

**6.175.3.6** `virtual std::string decaf::security::cert::Certificate::toString () const`  
[pure virtual]

Returns a string representation of this certificate.

**Returns:**

a string representation of this certificate

**6.175.3.7** `virtual void decaf::security::cert::Certificate::verify (const  
PublicKey & publicKey, const std::string & sigProvider) const  
throw ( NoSuchAlgorithmException, InvalidKeyException,  
NoSuchProviderException, SignatureException, CertificateException)`  
[pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key. Uses the verification engine of the specified provider.

**Parameters:**

*publicKey* The public key used to carry out the validation.

*sigProvider* The name of the signature provider

**Exceptions:**

*NoSuchAlgorithmException* (p. 2823) - on unsupported signature algorithms.

*InvalidKeyException* (p. 2132) - on incorrect key.

*NoSuchProviderException* (p. 2829) - if there's no default provider.

*SignatureException* (p. 3497) - on signature errors.

*CertificateException* (p. 1092) - on encoding errors.

**6.175.3.8** `virtual void decaf::security::cert::Certificate::verify (const  
PublicKey & publicKey) const throw ( NoSuchAlgorithmException,  
InvalidKeyException, NoSuchProviderException, SignatureException,  
CertificateException)` [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

**Parameters:**

*publicKey* The public key used to carry out the validation.

**Exceptions:**

*NoSuchAlgorithmException* (p. 2823) - on unsupported signature algorithms.

*InvalidKeyException* (p. 2132) - on incorrect key.

*NoSuchProviderException* (p. 2829) - if there's no default provider.

*SignatureException* (p. 3497) - on signature errors.

*CertificateException* (p. 1092) - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**



## 6.176 decaf::security::cert::CertificateEncodingException Class Reference

#include <src/main/decaf/security/cert/CertificateEncodingException.h> Inheritance diagram for decaf::security::cert::CertificateEncodingException:

### Public Member Functions

- **CertificateEncodingException** () throw ()  
*Default Constructor.*
- **CertificateEncodingException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateEncodingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateEncodingException** \* clone () const  
*Clones this exception.*
- virtual ~**CertificateEncodingException** () throw ()

### 6.176.1 Constructor & Destructor Documentation

**6.176.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException () throw () [inline]**

Default Constructor.

**6.176.1.2 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const Exception & ex) throw () [inline]**

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

**6.176.1.3 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & ex) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.176.1.4** `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.176.1.5** `virtual decaf::security::cert::CertificateEncodingException::~CertificateEncodingException () throw () [inline, virtual]`

**6.176.2 Member Function Documentation**

**6.176.2.1** `virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1093).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateEncodingException.h`

## 6.177 decaf::security::cert::CertificateException Class Reference

#include <src/main/decaf/security/cert/CertificateException.h> Inheritance diagram for decaf::security::cert::CertificateException:

### Public Member Functions

- **CertificateException** () throw ()  
*Default Constructor.*
- **CertificateException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateException** (const **CertificateException** &ex) throw ()  
*Copy Constructor.*
- **CertificateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateException** () throw ()

### 6.177.1 Constructor & Destructor Documentation

#### 6.177.1.1 decaf::security::cert::CertificateException::CertificateException () throw () [inline]

Default Constructor.

#### 6.177.1.2 decaf::security::cert::CertificateException::CertificateException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

#### 6.177.1.3 decaf::security::cert::CertificateException::CertificateException (const CertificateException & ex) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.177.1.4** `decaf::security::cert::CertificateException::CertificateException (const char * file, const int lineNumber, const char * msg, ...) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.177.1.5** `virtual decaf::security::cert::CertificateException::~~CertificateException () throw ()` [inline, virtual]

**6.177.2 Member Function Documentation**

**6.177.2.1** `virtual CertificateException* decaf::security::cert::CertificateException::clone () const`  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1973).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 1091), `decaf::security::cert::CertificateExpiredException` (p. 1095), `decaf::security::cert::CertificateNotYetValidException` (p. 1097), and `decaf::security::cert::CertificateParsingException` (p. 1099).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

## 6.178 decaf::security::cert::CertificateExpiredException Class Reference

#include <src/main/decaf/security/cert/CertificateExpiredException.h>Inheritance diagram for decaf::security::cert::CertificateExpiredException:

### Public Member Functions

- **CertificateExpiredException** () throw ()  
*Default Constructor.*
- **CertificateExpiredException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()  
*Copy Constructor.*
- **CertificateExpiredException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateExpiredException** \* clone () const  
*Clones this exception.*
- virtual ~**CertificateExpiredException** () throw ()

### 6.178.1 Constructor & Destructor Documentation

#### 6.178.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException () throw () [inline]

Default Constructor.

#### 6.178.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

#### 6.178.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const CertificateExpiredException & ex) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.178.1.4** `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.178.1.5** `virtual decaf::security::cert::CertificateExpiredException::~~CertificateExpiredException () throw () [inline, virtual]`

**6.178.2 Member Function Documentation**

**6.178.2.1** `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1093).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

## 6.179 decaf::security::cert::CertificateNotYetValidException Class Reference

#include <src/main/decaf/security/cert/CertificateNotYetValidException.h> Inheritance diagram for decaf::security::cert::CertificateNotYetValidException:

### Public Member Functions

- **CertificateNotYetValidException** () throw ()  
*Default Constructor.*
- **CertificateNotYetValidException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()  
*Copy Constructor.*
- **CertificateNotYetValidException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateNotYetValidException** \* clone () const  
*Clones this exception.*
- virtual ~**CertificateNotYetValidException** () throw ()

### 6.179.1 Constructor & Destructor Documentation

**6.179.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException () throw () [inline]**

Default Constructor.

**6.179.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const Exception & ex) throw () [inline]**

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

**6.179.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & ex) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.179.1.4** `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.179.1.5** `virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException () throw ()` [inline, virtual]

**6.179.2 Member Function Documentation**

**6.179.2.1** `virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone () const` [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1093).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateNotYetValidException.h`



## 6.180 decaf::security::cert::CertificateParsingException Class Reference

#include <src/main/decaf/security/cert/CertificateParsingException.h>Inheritance diagram for decaf::security::cert::CertificateParsingException:

### Public Member Functions

- **CertificateParsingException** () throw ()  
*Default Constructor.*
- **CertificateParsingException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateParsingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateParsingException** \* clone () const  
*Clones this exception.*
- virtual ~**CertificateParsingException** () throw ()

### 6.180.1 Constructor & Destructor Documentation

#### 6.180.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () [inline]

Default Constructor.

#### 6.180.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

#### 6.180.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & ex) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.180.1.4** `decaf::security::cert::CertificateParsingException::CertificateParsingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.180.1.5** `virtual decaf::security::cert::CertificateParsingException::~~CertificateParsingException () throw () [inline, virtual]`

**6.180.2 Member Function Documentation**

**6.180.2.1** `virtual CertificateParsingException* decaf::security::cert::CertificateParsingException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.1093).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateParsingException.h`

## 6.181 decaf::lang::Character Class Reference

#include <src/main/decaf/lang/Character.h> Inheritance diagram for decaf::lang::Character:

### Public Member Functions

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const  
*Compares this **Character** (p. 1100) instance with another.*
- virtual bool **operator==** (const **Character** &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Character** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const char &c) const  
*Compares this **Character** (p. 1100) instance with a char type.*
- virtual bool **operator==** (const char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static **Character valueOf** (char value)  
*Returns a **Character** (p. 1100) instance representing the specified char value.*
- static bool **isWhitespace** (char c)  
*Indicates whether or not the given character is considered whitespace.*
- static bool **isDigit** (char c)  
*Indicates whether or not the given character is a digit.*
- static bool **isLowerCase** (char c)  
*Indicates whether or not the given character is a lower case character.*
- static bool **isUpperCase** (char c)  
*Indicates whether or not the given character is a upper case character.*
- static bool **isLetter** (char c)  
*Indicates whether or not the given character is a letter.*
- static bool **isLetterOrDigit** (char c)  
*Indicates whether or not the given character is either a letter or a digit.*
- static bool **isISOControl** (char c)  
*Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.*
- static int **digit** (char c, int radix)  
*Returns the numeric value of the character ch in the specified radix.*

## Static Public Attributes

- static const int **MIN\_RADIX** = 2  
*The minimum radix available for conversion to and from strings.*
- static const int **MAX\_RADIX** = 36  
*The maximum radix available for conversion to and from strings.*
- static const char **MIN\_VALUE** = (char)0x7F  
*The minimum value that a signed char can take on.*
- static const char **MAX\_VALUE** = (char)0x80  
*The maximum value that a signed char can take on.*
- static const int **SIZE** = 8  
*The size of the primitive charactor in bits.*

## 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 decaf::lang::Character::Character (char *value*)

#### Parameters:

*value* - char to wrap.

## 6.181.2 Member Function Documentation

### 6.181.2.1 virtual unsigned char decaf::lang::Character::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2835).

### 6.181.2.2 virtual int decaf::lang::Character::compareTo (const char & *c*) const [inline, virtual]

Compares this **Character** (p. 1100) instance with a char type.

#### Parameters:

*c* - the char instance to be compared

#### Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< char >** (p. 1214).

### 6.181.2.3 virtual int decaf::lang::Character::compareTo (const Character & *c*) const [inline, virtual]

Compares this **Character** (p. 1100) instance with another.

#### Parameters:

*c* - the **Character** (p. 1100) instance to be compared

#### Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.181.2.4 static int decaf::lang::Character::digit (char *c*, int *radix*) [static]**

Returns the numeric value of the character *ch* in the specified radix. If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

\* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. \* The character is one of the uppercase Latin letters 'A' through 'Z' and its **code** (p. 1183) is less than `radix + 'A' - 10`. In this case, `ch - 'A' + 10` is returned. \* The character is one of the lowercase Latin letters 'a' through 'z' and its **code** (p. 1183) is less than `radix + 'a' - 10`. In this case, `ch - 'a' + 10` is returned.

**Parameters:**

*c* - the char to be converted  
*radix* - the radix of the number

**Returns:**

the numeric value of the number represented in the given radix

**6.181.2.5 virtual double decaf::lang::Character::doubleValue () const [inline, virtual]**

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.181.2.6 bool decaf::lang::Character::equals (const char & *c*) const [inline, virtual]****Returns:**

true if the two Characters have the same value.

Implements **decaf::lang::Comparable< char >** (p. 1215).

**6.181.2.7 bool decaf::lang::Character::equals (const Character & *c*) const [inline]****Returns:**

true if the two **Character** (p. 1100) Objects have the same value.

**6.181.2.8 virtual float decaf::lang::Character::floatValue () const [inline, virtual]**

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.181.2.9** `virtual int decaf::lang::Character::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.181.2.10** `static bool decaf::lang::Character::isDigit (char c) [inline, static]`

Indicates whether or not the given character is a digit.

**6.181.2.11** `static bool decaf::lang::Character::isISOControl (char c) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

**Parameters:**

*c* - the character, including supplementary characters

**Returns:**

true if the char is an ISO control character

**6.181.2.12** `static bool decaf::lang::Character::isLetter (char c) [inline, static]`

Indicates whether or not the given character is a letter.

**6.181.2.13** `static bool decaf::lang::Character::isLetterOrDigit (char c) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

**6.181.2.14** `static bool decaf::lang::Character::isLowerCase (char c) [inline, static]`

Indicates whether or not the given character is a lower case character.

**6.181.2.15** `static bool decaf::lang::Character::isUpperCase (char c) [inline, static]`

Indicates whether or not the given character is a upper case character.

**6.181.2.16** `static bool decaf::lang::Character::isWhitespace (char c) [inline, static]`

Indicates whether or not the given character is considered whitespace.

**6.181.2.17** `virtual long long decaf::lang::Character::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.181.2.18** `virtual bool decaf::lang::Character::operator< (const char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1215).

**6.181.2.19** `virtual bool decaf::lang::Character::operator< (const Character & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.181.2.20** `virtual bool decaf::lang::Character::operator== (const char & c) const [inline, virtual]`

Compares equality between this object and the one passed.

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1215).



**6.181.2.21** `virtual bool decaf::lang::Character::operator==(const Character & c) const [inline, virtual]`

Compares equality between this object and the one passed.

**Parameters:**

*c* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.181.2.22** `virtual short decaf::lang::Character::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2837).

**6.181.2.23** `std::string decaf::lang::Character::toString () const`

**Returns:**

this **Character** (p. 1100) Object as a **String** (p. 3665) Representation

**6.181.2.24** `static Character decaf::lang::Character::valueOf (char value) [inline, static]`

Returns a **Character** (p. 1100) instance representing the specified char value.

**Parameters:**

*value* - the primitive char to wrap.

**Returns:**

a new Character instance that wraps this value.

## 6.181.3 Field Documentation

**6.181.3.1** `const int decaf::lang::Character::MAX_RADIX = 36 [static]`

The maximum radix available for conversion to and from strings.

**6.181.3.2** `const char decaf::lang::Character::MAX_VALUE = (char)0x80 [static]`

The maximum value that a signed char can take on.

**6.181.3.3** `const int decaf::lang::Character::MIN_RADIX = 2` [static]

The minimum radix available for conversion to and from strings.

**6.181.3.4** `const char decaf::lang::Character::MIN_VALUE = (char)0x7F` [static]

The minimum value that a signed char can take on.

**6.181.3.5** `const int decaf::lang::Character::SIZE = 8` [static]

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

## 6.182 decaf::internal::nio::CharArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/CharArrayBuffer.h> Inheritance diagram for decaf::internal::nio::CharArrayBuffer:

### Public Member Functions

- **CharArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **CharArrayBuffer** (p. 1108) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **CharArrayBuffer** (char \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **CharArrayBuffer** (p. 1108) object that wraps the given array.*

- **CharArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **CharArrayBuffer** (const CharArrayBuffer &other)

*Create a **CharArrayBuffer** (p. 1108) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**CharArrayBuffer** ()
- virtual char \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the character array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928).*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset into the backing array where index zero starts.*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual CharBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only char buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only char buffer which the caller then owns.*

- virtual CharBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns:**

*a reference to this **CharBuffer** (p. 1119).*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) - If this buffer is read-only

- virtual CharBuffer \* **duplicate** ()

*Creates a new char buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

*The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*a new char **Buffer** (p. 928) which the caller owns.*

- virtual char **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the character at this buffer's current position, and then increments the position.*

**Returns:**

*the char at the current position.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return

- virtual char **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the char at the given index.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the char is to be read.

**Returns:**

the char that is located at the given index.

**Exceptions:**

**IndexOutOfBoundsException** if *index* is not smaller than the buffer's limit or is negative.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

- virtual CharBuffer & **put** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given char into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The char value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual CharBuffer & **put** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given char into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The char to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if *index* greater than the buffer's limit minus the size of the type being written, or *index* is negative.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual CharBuffer \* **slice** () const

Creates a new **CharBuffer** (p. 1119) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **CharBuffer** (p. 1119) which the caller owns.

- virtual **lang::CharSequence \* subSequence** (int start, int end) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 932) + start, and its limit will be **position()** (p. 932) + end. The new **Buffer** (p. 928) will be read-only if, and only if, this buffer is read-only.

**Parameters:**

**start** The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 933).

**end** The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 933).

**Returns:**

The new character buffer, caller owns.

**Exceptions:**

**IndexOutOfBoundsException** if the preconditions on start and end fail.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **CharArrayBuffer** (p. 1108) as Read-Only.

## Protected Attributes

- bool **readOnly**
- decaf::lang::Pointer< ByteArrayAdapter > **\_array**
- int **offset**
- int **length**

## 6.182.1 Constructor & Destructor Documentation

### 6.182.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (int *size*, bool *readOnly* = false) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **CharArrayBuffer** (p.1108) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

#### Parameters:

*size* The size of the array, this is the limit we read and write to.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*IllegalArgumentException* if the capacity value is negative.

### 6.182.1.2 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char \* *array*, int *size*, int *offset*, int *length*, bool *readOnly* = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **CharArrayBuffer** (p.1108) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

#### Parameters:

*array* The actual array to wrap.

*size* The size of the given array.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

### 6.182.1.3 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & *array*, int *offset*, int *length*, bool *readOnly* = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset. The capacity and limit of the new **CharArrayBuffer** (p.1108) will be that of the remaining capacity of the passed buffer.

**Parameters:**

- array* The ByteArrayAdapter to wrap.
- offset* The position that is this buffers start position.
- length* The limit of how many bytes into the array this Buffer can write.
- readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

- NullPointerException* if array is NULL
- IndexOutOfBoundsException* if offset + length is greater than array size.

#### 6.182.1.4 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & *other*)

Create a **CharArrayBuffer** (p. 1108) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

**Parameters:**

- other* The **CharArrayBuffer** (p. 1108) this one is to mirror.

#### 6.182.1.5 virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer () [virtual]

### 6.182.2 Member Function Documentation

#### 6.182.2.1 virtual char\* decaf::internal::nio::CharArrayBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

- the array that backs this **Buffer** (p. 928).

**Exceptions:**

- ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.
- UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1124).



**6.182.2.2** `virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1124).

**6.182.2.3** `virtual CharBuffer* de-  
caf::internal::nio::CharArrayBuffer::asReadOnlyBuffer ()  
const [virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 1125).

**6.182.2.4** `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ()  
throw ( decaf::nio::ReadOnlyBufferException ) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 932) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 932) - 1 is copied to index `n = limit()` (p. 932) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **CharBuffer** (p. 1119).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 1125).

**6.182.2.5 virtual CharBuffer\* decaf::internal::nio::CharArrayBuffer::duplicate ()**  
[virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new char **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1126).

**6.182.2.6 virtual char decaf::internal::nio::CharArrayBuffer::get (int *index*) const**  
**throw ( lang::exceptions::IndexOutOfBoundsException )** [virtual]

Absolute get method.

Reads the char at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the char is to be read.

**Returns:**

the char that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit or is negative.

Implements **decaf::nio::CharBuffer** (p. 1127).

**6.182.2.7 virtual char decaf::internal::nio::CharArrayBuffer::get () throw (**  
**decaf::nio::BufferUnderflowException )** [virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

**Returns:**

the char at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 1127).

**6.182.2.8 virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const**  
[inline, virtual]

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 1128).

**6.182.2.9 virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const**  
[inline, virtual]

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 931).

**6.182.2.10 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put**  
(int *index*, char *value*) throw ( de-  
cafe::lang::exceptions::IndexOutOfBoundsException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given char into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The char to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1129).

**6.182.2.11** **virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (char *value*) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )** [virtual]

Writes the given char into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The char value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1130).

**6.182.2.12** **virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool *value*)** [inline, protected, virtual]

Sets this **CharArrayBuffer** (p. 1108) as Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.182.2.13** **virtual CharBuffer\* decaf::internal::nio::CharArrayBuffer::slice () const** [virtual]

Creates a new **CharBuffer** (p. 1119) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **CharBuffer** (p. 1119) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1132).

**6.182.2.14** `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (int start, int end) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 932) + *start*, and its limit will be **position()** (p. 932) + *end*. The new **Buffer** (p. 928) will be read-only if, and only if, this buffer is read-only.

#### Parameters:

***start*** The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 933).

***end*** The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than *start* and no larger than **remaining()** (p. 933).

#### Returns:

The new character buffer, caller owns.

#### Exceptions:

***IndexOutOfBoundsException*** if the preconditions on *start* and *end* fail.

Implements **decaf::nio::CharBuffer** (p. 1132).

### 6.182.3 Field Documentation

**6.182.3.1** `decaf::lang::Pointer<ByteArrayAdapter> decaf::internal::nio::CharArrayBuffer::_array` [protected]

**6.182.3.2** `int decaf::internal::nio::CharArrayBuffer::length` [protected]

**6.182.3.3** `int decaf::internal::nio::CharArrayBuffer::offset` [protected]

**6.182.3.4** `bool decaf::internal::nio::CharArrayBuffer::readOnly` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/CharArrayBuffer.h`

## 6.183 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:.

```
#include <src/main/decaf/nio/CharBuffer.h>
Inheritance diagram for decaf::nio::CharBuffer:
```

### Public Member Functions

- virtual **~CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer & append** (char value) throw ( BufferOverflowException, ReadOnlyBufferException )  
*Appends the specified character to this buffer.*
- **CharBuffer & append** (const lang::CharSequence \*value) throw ( BufferOverflowException, ReadOnlyBufferException )  
*Appends the specified character sequence to this buffer.*
- **CharBuffer & append** (const lang::CharSequence \*value, int start, int end) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException )  
*Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.*
- virtual char \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the character array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **CharBuffer \* asReadOnlyBuffer** () const =0  
*Creates a new, read-only char buffer that shares this buffer's content.*
- char **charAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads the character at the given index relative to the current position.*
- virtual **CharBuffer & compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **CharBuffer \* duplicate** ()=0  
*Creates a new char buffer that shares this buffer's content.*
- virtual char **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*

- virtual char **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **CharBuffer** & **get** (std::vector< char > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **CharBuffer** & **get** (char \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible char array.*
- int **length** () const  
*Returns the length of this character buffer.*
- **CharBuffer** & **put** (**CharBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the chars remaining in the given source buffer into this buffer.*
- **CharBuffer** & **put** (const char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers chars into this buffer from the given source array.*
- **CharBuffer** & **put** (std::vector< char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source char array into this buffer.*
- virtual **CharBuffer** & **put** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given char into this buffer at the current position, and then increments the position.*
- virtual **CharBuffer** & **put** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given char into this buffer at the given index.*
- **CharBuffer** & **put** (std::string &src, int start, int end) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Relative bulk put method (optional operation).*
- **CharBuffer** & **put** (const std::string &src) throw ( BufferOverflowException, ReadOnlyBufferException )  
*Relative bulk put method (optional operation).*

- virtual int **read** (**CharBuffer** \*target) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

- virtual **lang::CharSequence** \* **subSequence** (int start, int end) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.*

- virtual **CharBuffer** \* **slice** () const =0

*Creates a new **CharBuffer** (p. 1119) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **CharBuffer** &value) const

- virtual bool **equals** (const **CharBuffer** &value) const

- virtual bool **operator==** (const **CharBuffer** &value) const

- virtual bool **operator<** (const **CharBuffer** &value) const

## Static Public Member Functions

- static **CharBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new character buffer.*

- static **CharBuffer** \* **wrap** (char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **CharBuffer** (p. 1119).*

- static **CharBuffer** \* **wrap** (std::vector< char > &buffer)

*Wraps the passed STL char Vector in a **CharBuffer** (p. 1119).*

## Protected Member Functions

- **CharBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **CharBuffer** (p. 1119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*



### 6.183.1 Detailed Description

This class defines four categories of operations upon character buffers:.

- o Absolute and relative get and put methods that read and write single characters;
- o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer.
- o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the `CharSequence` interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package `decaf.util.regex`.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

### 6.183.2 Constructor & Destructor Documentation

#### 6.183.2.1 `decaf::nio::CharBuffer::CharBuffer (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **CharBuffer** (p. 1119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

##### Parameters:

***capacity*** The size of the array, this is the limit we read and write to.

##### Exceptions:

***IllegalArgumentException*** if capacity is negative.

#### 6.183.2.2 `virtual decaf::nio::CharBuffer::~~CharBuffer ()` [inline, virtual]

### 6.183.3 Member Function Documentation

#### 6.183.3.1 `static CharBuffer* decaf::nio::CharBuffer::allocate (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Allocates a new character buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

##### Parameters:

***capacity*** The size of the Char buffer in chars ( 1 byte ).

**Returns:**

the **CharBuffer** (p. 1119) that was allocated, caller owns.

**Exceptions:**

*IndexOutOfBoundsException* if capacity is negative.

**6.183.3.2 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence \* *value*, int *start*, int *end*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException ) [virtual]**

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

**Parameters:**

*value* The CharSequence to append.

*start* The index to start appending from.

*end* The index to append to.

**Returns:**

a reference to this modified **CharBuffer** (p. 1119).

**Exceptions:**

*BufferOverflowException* (p. 954) if there is no more space

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*IndexOutOfBoundsException* if start > end, or > length of sequence.

Implements **decaf::lang::Appendable** (p. 726).

**6.183.3.3 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence \* *value*) throw ( BufferOverflowException, ReadOnlyBufferException ) [virtual]**

Appends the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

**Parameters:**

*value* The CharSequence to append.

**Returns:**

a reference to this modified **CharBuffer** (p. 1119)

**Exceptions:**

*BufferOverflowException* (p. 954) if there is no more space

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

Implements **decaf::lang::Appendable** (p. 726).

#### 6.183.3.4 CharBuffer& decaf::nio::CharBuffer::append (char *value*) throw ( BufferOverflowException, ReadOnlyBufferException ) [virtual]

Appends the specified character to this buffer.

##### Parameters:

*value* The char to append.

##### Returns:

a reference to this modified **CharBuffer** (p. 1119).

##### Exceptions:

*BufferOverflowException* (p. 954) if there is no more space

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

Implements **decaf::lang::Appendable** (p. 726).

#### 6.183.3.5 virtual char\* decaf::nio::CharBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the character array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

##### Returns:

the array that backs this **Buffer** (p. 928).

##### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1113).

#### 6.183.3.6 virtual int decaf::nio::CharBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

##### Returns:

The offset into the backing array where index zero starts.

##### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 1113).

#### 6.183.3.7 `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const` [pure virtual]

Creates a new, read-only char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns:

The new, read-only char buffer which the caller then owns.

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 1114).

#### 6.183.3.8 `char decaf::nio::CharBuffer::charAt (int index) const throw (` `decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads the character at the given index relative to the current position.

#### Parameters:

*index* - The index of the character to be read relative to position

#### Returns:

The character at index `position()` (p. 932) + index.

#### Exceptions:

*IndexOutOfBoundsException* if the index + the current position exceeds the size of the buffer or the index is negative.

Implements `decaf::lang::CharSequence` (p. 1135).

#### 6.183.3.9 `virtual CharBuffer& decaf::nio::CharBuffer::compact () throw (` `ReadOnlyBufferException )` [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 932) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 932) - 1 is copied to index `n = limit()` (p. 932) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **CharBuffer** (p. 1119).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1114).

**6.183.3.10** `virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & value) const` [virtual]

**6.183.3.11** `virtual CharBuffer* decaf::nio::CharBuffer::duplicate ()` [pure virtual]

Creates a new char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new char **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1114).

**6.183.3.12** `virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & value) const` [virtual]

**6.183.3.13** `CharBuffer& decaf::nio::CharBuffer::get (char * buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Relative bulk get method. This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies `length` chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

**Parameters:**

*buffer* The pointer to an allocated buffer to fill.

*size* The size of the buffer passed.

*offset* The position in the buffer to start filling.

*length* The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* ( p. 957) if there are fewer than length chars remaining in this buffer

*NullPointerException* if the passed buffer is null.

*IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

#### 6.183.3.14 CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > buffer) throw ( BufferUnderflowException )

Relative bulk get method. This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize( N ) before calling this get method.

**Returns:**

a reference to this **CharBuffer** (p. 1119).

**Exceptions:**

*BufferUnderflowException* ( p. 957) if there are fewer than length chars remaining in this buffer.

#### 6.183.3.15 virtual char decaf::nio::CharBuffer::get (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Absolute get method. Reads the char at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the char is to be read.

**Returns:**

the char that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit or is negative.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1115).

#### 6.183.3.16 virtual char decaf::nio::CharBuffer::get () throw ( BufferUnderflowException ) [pure virtual]

Relative get method. Reads the character at this buffer's current position, and then increments the position.

**Returns:**

the char at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1115).

**6.183.3.17 virtual bool decaf::nio::CharBuffer::hasArray () const [pure virtual]**

Tells whether or not this buffer is backed by an accessible char array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1115).

**6.183.3.18 int decaf::nio::CharBuffer::length () const [inline, virtual]**

Returns the length of this character buffer.

**Returns:**

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 1136).

**6.183.3.19 virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & value) const [virtual]****6.183.3.20 virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & value) const [virtual]****6.183.3.21 CharBuffer& decaf::nio::CharBuffer::put (const std::string & src) throw ( BufferOverflowException, ReadOnlyBufferException )**

Relative bulk put method (optional operation). This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form dst.put(s) behaves in exactly the same way as the invocation.

**Parameters:**

*src* The string to copy from.

**Returns:**

a reference to this **CharBuffer** (p. 1119).

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

### 6.183.3.22 CharBuffer& decaf::nio::CharBuffer::put (std::string & *src*, int *start*, int *end*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException )

Relative bulk put method (optional operation). This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if *end* - *start* > **remaining()** (p. 933), then no characters are transferred and a **BufferOverflowException** (p. 954) is thrown.

#### Returns:

a reference to this buffer

Otherwise, this method copies *n* = *end* - *start* characters from the given string into this buffer, starting at the given *start* index and at the current position of this buffer. The position of this buffer is then incremented by *n*.

#### Parameters:

*src* The string to copy from.  
*start* The position in *src* to start from.  
*end* The position in *src* to stop at.

#### Returns:

a reference to this **CharBuffer** (p. 1119).

#### Exceptions:

**BufferOverflowException** (p. 954) if this buffer's current position is not  
**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

### 6.183.3.23 virtual CharBuffer& decaf::nio::CharBuffer::put (int *index*, char *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes the given char into this buffer at the given index.

#### Parameters:

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The char to write.

#### Returns:

a reference to this buffer.

#### Exceptions:

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1116).



**6.183.3.24** virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The char value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1116).

**6.183.3.25** CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source char array into this buffer. This is the same as calling put( &buffer[0], 0, buffer.size()).

**Parameters:**

*buffer* The buffer whose contents are copied to this **CharBuffer** (p. 1119).

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

**6.183.3.26** CharBuffer& decaf::nio::CharBuffer::put (const char \* *buffer*, int *size*, int *offset*, int *length*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

This method transfers chars into this buffer from the given source array. If there are more chars to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 933), then no chars are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

**Parameters:**

*buffer* The array from which chars are to be read.  
*size* The size of the buffer passed.  
*offset* The offset within the array of the first char to be read.  
*length* The number of chars to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only  
**NullPointerException** if the passed buffer is null.  
**IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

**6.183.3.27** `CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & src)  
 throw ( BufferOverflowException, ReadOnlyBufferException,  
 decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the chars remaining in the given source buffer into this buffer. If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 933), then no chars are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

**Parameters:**

*src* - the buffer to take chars from an place in this one.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer for the remaining chars in the source buffer.  
**IllegalArgumentException** if the source buffer is this buffer.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**6.183.3.28** `virtual int decaf::nio::CharBuffer::read (CharBuffer * target)  
 throw ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IllegalArgumentException,  
 ReadOnlyBufferException ) [virtual]`

Attempts to read characters into the specified character buffer. The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

**Parameters:**

*target* The buffer to read characters into

**Returns:**

The number of characters added to the buffer, or string::npos if this source of characters is at its end

**Exceptions:**

*NullPointerException* if target is Null.

*IllegalArgumentException* if target is this **CharBuffer** (p. 1119).

*ReadOnlyBufferException* (p. 3169) if this buffer is in read-only mode.

### 6.183.3.29 virtual CharBuffer\* decaf::nio::CharBuffer::slice () const [pure virtual]

Creates a new **CharBuffer** (p. 1119) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **CharBuffer** (p. 1119) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1117).

### 6.183.3.30 virtual lang::CharSequence\* decaf::nio::CharBuffer::subSequence (int start, int end) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position. The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 932) + start, and its limit will be **position()** (p. 932) + end. The new **Buffer** (p. 928) will be read-only if, and only if, this buffer is read-only.

**Parameters:**

*start* The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 933).

*end* The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 933).

**Returns:**

The new character buffer, caller owns.

**Exceptions:**

***IndexOutOfBoundsException*** if the preconditions on start and end fail.

Implements **decaf::lang::CharSequence** (p.1136).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p.1117).

**6.183.3.31 virtual std::string decaf::nio::CharBuffer::toString () const [virtual]****Returns:**

a std::string describing this object

Implements **decaf::lang::CharSequence** (p.1136).

**6.183.3.32 static CharBuffer\* decaf::nio::CharBuffer::wrap (std::vector< char > & buffer) [static]**

Wraps the passed STL char Vector in a **CharBuffer** (p.1119). The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new **CharBuffer** (p.1119) that is backed by buffer, caller owns.

**6.183.3.33 static CharBuffer\* decaf::nio::CharBuffer::wrap (char \* array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]**

Wraps the passed buffer with a new **CharBuffer** (p.1119). The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*array* The array that will back the new buffer.

*size* The size of the array passed in.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new **CharBuffer** (p. 1119) that is backed by buffer, caller owns.

**Exceptions:**

*NullPointerException* if the array pointer is Null.

*IndexOutOfBoundsException* if capacity is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**CharBuffer.h**

## 6.184 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 1135) is a readable sequence of char values.

#include <src/main/decaf/lang/CharSequence.h> Inheritance diagram for decaf::lang::CharSequence:

### Public Member Functions

- virtual **~CharSequence** ()
- virtual int **length** () const =0
- virtual char **charAt** (int index) const =0 throw (lang::exceptions::IndexOutOfBoundsException )  
*Returns the Char at the specified index so long as the index is not greater than the length of the sequence.*
- virtual **CharSequence** \* **subSequence** (int start, int end) const =0 throw (lang::exceptions::IndexOutOfBoundsException )  
*Returns a new **CharSequence** (p. 1135) that is a subsequence of this sequence.*
- virtual std::string **toString** () const =0

### 6.184.1 Detailed Description

A **CharSequence** (p. 1135) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 1135) should implement comparable, it is therefore up to the dervied classes that implement this interface to define equality, which implies that comparison of two **CharSequences** does not have a contract on equality.

### 6.184.2 Constructor & Destructor Documentation

**6.184.2.1** virtual decaf::lang::CharSequence::~~CharSequence () [inline, virtual]

### 6.184.3 Member Function Documentation

**6.184.3.1** virtual char decaf::lang::CharSequence::charAt (int *index*) const throw (lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

#### Parameters:

*index* - position to return the char at.

#### Returns:

the char at the given position

**Exceptions:**

*IndexOutOfBoundsException* if index is > than **length()** (p. 1136) or negative

Implemented in **decaf::lang::String** (p. 3666), and **decaf::nio::CharBuffer** (p. 1125).

**6.184.3.2 virtual int decaf::lang::CharSequence::length () const [pure virtual]****Returns:**

the length of the underlying character sequence.

Implemented in **decaf::lang::String** (p. 3667), and **decaf::nio::CharBuffer** (p. 1128).

**6.184.3.3 virtual CharSequence\* decaf::lang::CharSequence::subSequence (int start, int end) const throw ( lang::exceptions::IndexOutOfBoundsException ) [pure virtual]**

Returns a new **CharSequence** (p. 1135) that is a subsequence of this sequence. The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

**Parameters:**

*start* - the start index, inclusive

*end* - the end index, exclusive

**Returns:**

a new **CharSequence** (p. 1135)

**Exceptions:**

*IndexOutOfBoundsException* if start or end > **length()** (p.1136) or start or end are negative.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p.1117), **decaf::lang::String** (p. 3667), and **decaf::nio::CharBuffer** (p. 1132).

**6.184.3.4 virtual std::string decaf::lang::CharSequence::toString () const [pure virtual]****Returns:**

the string representation of this **CharSequence** (p. 1135)

Implemented in **decaf::lang::String** (p. 3667), and **decaf::nio::CharBuffer** (p. 1133).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

## 6.185 decaf::util::zip::Checksum Class Reference

An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1142) of the bytes read, the **Checksum** (p. 1142) can then be used to verify the integrity of the input stream.

#include <src/main/decaf/util/zip/Checksum.h> Inheritance diagram for decaf::util::zip::Checksum:

### Public Member Functions

- **Checksum** (`InputStream *inputStream, Checksum *sum, bool own=false`)

Create a new instance of a **Checksum** (p. 1137).

- virtual `~Checksum()`
- `Checksum * getChecksum() const`

Returns a Pointer to the **Checksum** (p. 1142) that is in use by this **Checksum** (p. 1137).

- virtual long long **skip** (long long num) throw ( decaf::io::IOException )

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters:**

*num* The number of bytes to skip.

**Returns:**

total bytes skipped

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

### Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

#### 6.185.1 Detailed Description

An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1142) of the bytes read, the **Checksum** (p. 1142) can then be used to verify the integrity of the input stream.



Since:

1.0

## 6.185.2 Constructor & Destructor Documentation

### 6.185.2.1 decaf::util::zip::CheckedInputStream::CheckedInputStream (InputStream \* *inputStream*, Checksum \* *sum*, bool *own* = false)

Create a new instance of a **CheckedInputStream** (p. 1137).

Parameters:

*inputStream* The InputStream instance to Wrap.

*sum* The **Checksum** (p. 1142) instance to use (does not take ownership of the Pointer).

*own* Indicates if this filer should take ownership of the InputStream.

Exceptions:

*NullPointerException* if the **Checksum** (p. 1142) pointer is NULL.

### 6.185.2.2 virtual decaf::util::zip::CheckedInputStream::~~CheckedInputStream () [virtual]

## 6.185.3 Member Function Documentation

### 6.185.3.1 virtual int decaf::util::zip::CheckedInputStream::doReadArrayBounded (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

### 6.185.3.2 virtual int decaf::util::zip::CheckedInputStream::doReadByte () throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

### 6.185.3.3 Checksum\* decaf::util::zip::CheckedInputStream::getChecksum () const [inline]

Returns a Pointer to the **Checksum** (p. 1142) that is in use by this **CheckedInputStream** (p. 1137).

Returns:

the pointer to the **Checksum** (p. 1142) instance that is in use by this object.

#### 6.185.3.4 **virtual long long decaf::util::zip::CheckedInputStream::skip (long long *num*) throw ( decaf::io::IOException ) [virtual]**

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

##### Parameters:

*num* The number of bytes to skip.

##### Returns:

total bytes skipped

##### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 1142).

Reimplemented from **decaf::io::FilterInputStream** (p. 1899).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedInputStream.h**

## 6.186 decaf::util::zip::CheckedOutputStream Class Reference

An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p.1142) of the bytes written, the **Checksum** (p.1142) can then be used to verify the integrity of the output stream.

#include <src/main/decaf/util/zip/CheckedOutputStream.h> Inheritance diagram for decaf::util::zip::CheckedOutputStream:

### Public Member Functions

- **CheckedOutputStream** (decaf::io::OutputStream \*outputStream, Checksum \*sum, bool own=false)  
*Create a new instance of a **CheckedOutputStream** (p.1140).*
- virtual ~**CheckedOutputStream** ()
- **Checksum** \* **getChecksum** () const

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

#### 6.186.1 Detailed Description

An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p.1142) of the bytes written, the **Checksum** (p.1142) can then be used to verify the integrity of the output stream.

**Since:**

1.0

#### 6.186.2 Constructor & Destructor Documentation

- ##### 6.186.2.1 decaf::util::zip::CheckedOutputStream::CheckedOutputStream (decaf::io::OutputStream \* outputStream, Checksum \* sum, bool own = false)

Create a new instance of a **CheckedOutputStream** (p.1140).

**Parameters:**

- outputStream* The `OutputStream` instance to Wrap.
- sum* The **Checksum** (p.1142) instance to use (does not take ownership of the Pointer).
- own* Indicates if this filer should take ownership of the `InputStream`.

**Exceptions:**

*NullPointerException* if the **Checksum** (p. 1142) pointer is NULL.

**6.186.2.2** virtual decaf::util::zip::ChecksumOutputStream::~ChecksumOutputStream  
( ) [virtual]

**6.186.3 Member Function Documentation**

**6.186.3.1** virtual void decaf::util::zip::ChecksumOutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.186.3.2** virtual void decaf::util::zip::ChecksumOutputStream::doWriteByte (unsigned char *value*) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.186.3.3** Checksum\* decaf::util::zip::ChecksumOutputStream::getChecksum () const [inline]

**Returns:**

a pointer to the **Checksum** (p. 1142) instance in use by this object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/ChecksumOutputStream.h

## 6.187 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p.1142) values in the Zip package.

#include <src/main/decaf/util/zip/Checksum.h> Inheritance diagram for decaf::util::zip::Checksum:

### Public Member Functions

- virtual **~Checksum** ()
- virtual long long **getValue** () const =0
- virtual void **reset** ()=0  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)=0  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)=0  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length)=0 throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)=0  
*Updates the current checksum with the specified byte value.*

### 6.187.1 Detailed Description

An interface used to represent **Checksum** (p.1142) values in the Zip package.

Since:

1.0

### 6.187.2 Constructor & Destructor Documentation

**6.187.2.1** virtual decaf::util::zip::Checksum::~~Checksum () [inline, virtual]

### 6.187.3 Member Function Documentation

**6.187.3.1** virtual long long decaf::util::zip::Checksum::getValue () const [pure virtual]

Returns:

the current checksum value.

Implemented in **decaf::util::zip::Adler32** (p. 723), and **decaf::util::zip::CRC32** (p. 1525).

#### 6.187.3.2 virtual void decaf::util::zip::Checksum::reset () [pure virtual]

Reset the checksum to its initial value.

Implemented in **decaf::util::zip::Adler32** (p. 723), and **decaf::util::zip::CRC32** (p. 1525).

#### 6.187.3.3 virtual void decaf::util::zip::Checksum::update (int *byte*) [pure virtual]

Updates the current checksum with the specified byte value.

##### Parameters:

*byte* The byte value to update the current **Checksum** (p. 1142) with (0..255).

Implemented in **decaf::util::zip::Adler32** (p. 723), and **decaf::util::zip::CRC32** (p. 1525).

#### 6.187.3.4 virtual void decaf::util::zip::Checksum::update (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Updates the current checksum with the specified array of bytes.

##### Parameters:

*buffer* The buffer to read the updated bytes from.

*size* The size of the passed buffer.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

##### Exceptions:

*NullPointerException* if the passed buffer is NULL.

*IndexOutOfBoundsException* if  $\text{offset} + \text{length} > \text{size of the buffer}$ .

Implemented in **decaf::util::zip::Adler32** (p. 723), and **decaf::util::zip::CRC32** (p. 1525).

#### 6.187.3.5 virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Updates the current checksum with the specified array of bytes.

##### Parameters:

*buffer* The buffer to read the updated bytes from.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

**Exceptions:**

*IndexOutOfBoundsException* if  $\text{offset} + \text{length} > \text{size of the buffer}$ .

Implemented in **decaf::util::zip::Adler32** (p. 724), and **decaf::util::zip::CRC32** (p. 1526).

**6.187.3.6 virtual void decaf::util::zip::Checksum::update (const std::vector< unsigned char > & *buffer*) [pure virtual]**

Updates the current checksum with the specified vector of bytes.

**Parameters:**

*buffer* The buffer to read the updated bytes from.

Implemented in **decaf::util::zip::Adler32** (p. 724), and **decaf::util::zip::CRC32** (p. 1526).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Checksum.h`

## 6.188 decaf::lang::exceptions::ClassCastException Class Reference

#include <src/main/decaf/lang/exceptions/ClassCastException.h> Inheritance diagram for decaf::lang::exceptions::ClassCastException:

### Public Member Functions

- **ClassCastException** () throw ()  
*Default Constructor.*
- **ClassCastException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()  
*Copy Constructor.*
- **ClassCastException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ClassCastException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ClassCastException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ClassCastException** \* clone () const  
*Clones this exception.*
- virtual ~**ClassCastException** () throw ()

### 6.188.1 Constructor & Destructor Documentation

#### 6.188.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException () throw () [inline]

Default Constructor.

#### 6.188.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.



**6.188.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.188.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.188.1.5 decaf::lang::exceptions::ClassCastException::ClassCastException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.188.1.6 decaf::lang::exceptions::ClassCastException::ClassCastException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.188.1.7** `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException  
() throw () [inline, virtual]`

## **6.188.2 Member Function Documentation**

**6.188.2.1** `virtual ClassCastException* de-  
caf::lang::exceptions::ClassCastException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/ClassCastException.h`

## 6.189 cms::Closeable Class Reference

Interface for a class that implements the close method.

#include <src/main/cms/Closeable.h> Inheritance diagram for cms::Closeable:

### Public Member Functions

- virtual `~Closeable ()`
- virtual void `close ()=0 throw ( CMSEException )`  
*Closes this object and deallocates the appropriate resources.*

### 6.189.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since:

1.0

### 6.189.2 Constructor & Destructor Documentation

**6.189.2.1** virtual cms::Closeable::~~Closeable () [inline, virtual]

### 6.189.3 Member Function Documentation

**6.189.3.1** virtual void cms::Closeable::close () throw ( CMSEException ) [pure virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

*CMSEException* (p. 1160) - If an error occurs while the resource is being closed.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1072), `activemq::cmsutil::CachedProducer` (p. 1076), `activemq::cmsutil::PooledSession` (p. 2957), `activemq::commands::ActiveMQTempDestination` (p. 577), `activemq::core::ActiveMQConnection` (p. 279), `activemq::core::ActiveMQConsumer` (p. 313), `activemq::core::ActiveMQProducer` (p. 472), `activemq::core::ActiveMQQueueBrowser` (p. 488), `activemq::core::ActiveMQSession` (p. 520), `cms::Connection` (p. 1263), and `cms::Session` (p. 3368).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

## 6.190 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

#include <src/main/decaf/io/Closeable.h> Inheritance diagram for decaf::io::Closeable:

### Public Member Functions

- virtual `~Closeable ()`
- virtual void `close ()=0 throw ( io::IOException )`  
*Closes this object and deallocates the appropriate resources.*

#### 6.190.1 Detailed Description

Interface for a class that implements the close method.

#### 6.190.2 Constructor & Destructor Documentation

**6.190.2.1** virtual `decaf::io::Closeable::~~Closeable ()` [inline, virtual]

#### 6.190.3 Member Function Documentation

**6.190.3.1** virtual void `decaf::io::Closeable::close () throw ( io::IOException )` [pure virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

#### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3292), `activemq::transport::failover::FailoverTransport` (p. 1876), `activemq::transport::inactivity::InactivityMonitor` (p. 2005), `activemq::transport::IOTransport` (p. 2147), `activemq::transport::mock::MockTransport` (p. 2770), `activemq::transport::tcp::TcpTransport` (p. 3753), `activemq::transport::TransportFilter` (p. 3893), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2902), `decaf::internal::io::StandardErrorOutputStream` (p. 3580), `decaf::internal::io::StandardOutputStream` (p. 3584), `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2863), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2883), `decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream` (p. 2886), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3748), `decaf::internal::net::tcp::TcpSocketOutputStream` (p. 3751), `decaf::io::BlockingByteArrayInputStream` (p. 841), `decaf::io::BufferedInputStream` (p. 937), `decaf::io::FilterInputStream` (p. 1897), `decaf::io::FilterOutputStream` (p. 1901), `decaf::io::InputStream` (p. 2045), `decaf::io::InputStreamReader` (p. 2054),

**decaf::io::OutputStream** (p. 2909), **decaf::io::OutputStreamWriter** (p. 2916), **decaf::net::Socket** (p. 3509), **decaf::util::logging::ConsoleHandler** (p. 1399), **decaf::util::logging::StreamHandler** (p. 3650), **decaf::util::zip::DeflaterOutputStream** (p. 1718), and **decaf::util::zip::InflaterInputStream** (p. 2039).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

## 6.191 activemq::transport::failover::CloseTransportsTask Class Reference

#include <src/main/activemq/transport/failover/CloseTransportsTask.h> Inheritance diagram for activemq::transport::failover::CloseTransportsTask:

### Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)  
*Add a new **Transport** (p. 3883) to close.*
- virtual bool **isPending** () const  
*This Task is pending if there are transports in the Queue that need to be closed.*
- virtual bool **iterate** ()  
*Return true until all transports have been closed and removed from the queue.*

### 6.191.1 Constructor & Destructor Documentation

- 6.191.1.1** **activemq::transport::failover::CloseTransportsTask::CloseTransportsTask** ()
- 6.191.1.2** **virtual**  
**activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask** () [virtual]

### 6.191.2 Member Function Documentation

- 6.191.2.1** **void** **activemq::transport::failover::CloseTransportsTask::add** (const **Pointer**< **Transport** > & *transport*)

Add a new **Transport** (p. 3883) to close.

- 6.191.2.2** **virtual bool** **activemq::transport::failover::CloseTransportsTask::isPending** () const [virtual]

This Task is pending if there are transports in the Queue that need to be closed.

#### Returns:

true if there is a **transport** (p. 97) in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 1221).

### 6.191.2.3 `virtual bool activemq::transport::failover::CloseTransportsTask::iterate ()` [virtual]

Return true until all transports have been closed and removed from the queue.

Implements `activemq::threads::Task` (p. 3734).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

## 6.192 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p.1170) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1324) to operate on.

#include <src/main/activemq/cmsutil/CmsAccessor.h> Inheritance diagram for activemq::cmsutil::CmsAccessor:

### Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** \* **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** \* **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (cms::ConnectionFactory \*connectionFactory)  
*Set the ConnectionFactory to use for obtaining CMS Connections.*
- virtual const cms::ConnectionFactory \* **getConnectionFactory** () const  
*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*
- virtual cms::ConnectionFactory \* **getConnectionFactory** ()  
*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*
- virtual void **setSessionAcknowledgeMode** (cms::Session::AcknowledgeMode sessionAcknowledgeMode)  
*Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.*
- virtual cms::Session::AcknowledgeMode **getSessionAcknowledgeMode** () const  
*Return the acknowledgment mode for CMS sessions.*

### Protected Member Functions

- **CmsAccessor** (const **CmsAccessor** &)
- **CmsAccessor** & **operator=** (const **CmsAccessor** &)
- virtual void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes this object and prepares it for use.*
- virtual void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Shuts down this object and destroys any allocated resources.*
- virtual cms::Connection \* **createConnection** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Create a CMS Connection via this template's ConnectionFactory.*



- virtual **cms::Session** \* **createSession** (**cms::Connection** \*con) throw ( **cms::CMSException**, **decaf::lang::exceptions::IllegalStateException** )  
*Create a CMS Session for the given Connection.*
- virtual void **checkConnectionFactory** () throw ( **decaf::lang::exceptions::IllegalStateException** )  
*Verifies that the connection factory is valid.*

### 6.192.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p.1170) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1324) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p.1157) adds further, destination-related properties.

Not intended to be used directly.

See also:

**activemq.cmsutil.CmsDestinationAccessor** (p.1157)  
**activemq.cmsutil.CmsTemplate** (p.1170)

### 6.192.2 Constructor & Destructor Documentation

- 6.192.2.1** **activemq::cmsutil::CmsAccessor::CmsAccessor** (const **CmsAccessor** &)  
[inline, protected]
- 6.192.2.2** **activemq::cmsutil::CmsAccessor::CmsAccessor** ()
- 6.192.2.3** **virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor** () [virtual]

### 6.192.3 Member Function Documentation

- 6.192.3.1** **virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory** () throw ( **decaf::lang::exceptions::IllegalStateException** ) [protected, virtual]

Verifies that the connection factory is valid.

- 6.192.3.2** **virtual cms::Connection\* activemq::cmsutil::CmsAccessor::createConnection** () throw ( **cms::CMSException**, **decaf::lang::exceptions::IllegalStateException** ) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

**Returns:**

the new CMS Connection

**Exceptions:**

**cms::CMSException** (p. 1160) if thrown by CMS API methods

**6.192.3.3** `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * con) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [protected, virtual]`

Create a CMS Session for the given Connection.

**Parameters:**

*con* the CMS Connection to create a Session for

**Returns:**

the new CMS Session

**Exceptions:**

*cms::CMSException* (p. 1160) if thrown by CMS API methods

**6.192.3.4** `virtual void activemq::cmsutil::CmsAccessor::destroy () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1158), and `activemq::cmsutil::CmsTemplate` (p.1173).

**6.192.3.5** `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

**6.192.3.6** `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

**6.192.3.7** `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const [inline, virtual]`

**6.192.3.8** `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () [inline, virtual]`

**6.192.3.9** `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const [inline, virtual]`

Return the acknowledgment mode for CMS sessions.

**Returns:**

the acknowledgment mode applied by this accessor

**6.192.3.10** `virtual void activemq::cmsutil::CmsAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Initializes this object and prepares it for use. This should be called before any other methods are called. This version does nothing.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1158), and `activemq::cmsutil::CmsTemplate` (p.1176).

**6.192.3.11** `CmsAccessor& activemq::cmsutil::CmsAccessor::operator= (const CmsAccessor &) [inline, protected]`

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1159), and `activemq::cmsutil::CmsTemplate` (p.1176).

**6.192.3.12** `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * connectionFactory) [inline, virtual]`

Set the ConnectionFactory to use for obtaining CMS Connections.

**6.192.3.13** `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode sessionAcknowledgeMode) [inline, virtual]`

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message. Default is `AUTO_ACKNOWLEDGE`.

**Parameters:**

*sessionAcknowledgeMode* the acknowledgment mode

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

## 6.193 `activemq::cmsutil::CmsDestinationAccessor` Class Reference

Extends the `CmsAccessor` (p. 1153) to add support for resolving destination names.

`#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>` Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:

### Public Member Functions

- `CmsDestinationAccessor ()`
- virtual `~CmsDestinationAccessor ()`
- virtual bool `isPubSubDomain ()` const
- virtual void `setPubSubDomain (bool pubSubDomain)`
- virtual `DestinationResolver * getDestinationResolver ()`
- virtual const `DestinationResolver * getDestinationResolver ()` const
- virtual void `setDestinationResolver (DestinationResolver *destRes)`

### Protected Member Functions

- `CmsDestinationAccessor (const CmsDestinationAccessor &)`
- `CmsDestinationAccessor & operator= (const CmsDestinationAccessor &)`
- virtual void `init ()` throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes the destination resolver.*
- virtual void `destroy ()` throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Calls `destroy()` (p. 1158) on the destination resolver.*
- virtual cms::Destination \* `resolveDestinationName (cms::Session *session, const std::string &destName)` throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Resolves the destination via the `DestinationResolver` (p. 1756).*
- virtual void `checkDestinationResolver ()` throw ( decaf::lang::exceptions::IllegalStateException )  
*Verifies that the destination resolver is valid.*

### 6.193.1 Detailed Description

Extends the `CmsAccessor` (p. 1153) to add support for resolving destination names. Not intended to be used directly.

See also:

`CmsTemplate` (p. 1170)  
`CmsAccessor` (p. 1153)

## 6.193.2 Constructor & Destructor Documentation

- 6.193.2.1** `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor (const CmsDestinationAccessor &) [inline, protected]`
- 6.193.2.2** `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ()`
- 6.193.2.3** `virtual  
activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ()  
[virtual]`

## 6.193.3 Member Function Documentation

- 6.193.3.1** `virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver ()  
throw ( decaf::lang::exceptions::IllegalStateException ) [protected,  
virtual]`

Verifies that the destination resolver is valid.

- 6.193.3.2** `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy () throw  
( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
[protected, virtual]`

Calls `destroy()` (p. 1158) on the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1155).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1173).

- 6.193.3.3** `virtual const DestinationResolver* ac-  
tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()  
const [inline, virtual]`

- 6.193.3.4** `virtual DestinationResolver* ac-  
tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()  
[inline, virtual]`

- 6.193.3.5** `virtual void activemq::cmsutil::CmsDestinationAccessor::init () throw  
( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
[protected, virtual]`

Initializes the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1156).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1176).

**6.193.3.6** virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const [inline, virtual]

**6.193.3.7** CmsDestinationAccessor& activemq::cmsutil::CmsDestinationAccessor::operator= (const CmsDestinationAccessor &) [inline, protected]

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 1156).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1176).

**6.193.3.8** virtual cms::Destination\* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session \* *session*, const std::string & *destName*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [protected, virtual]

Resolves the destination via the DestinationResolver (p. 1756).

#### Parameters:

*session* the session

*destName* the name of the destination.

#### Returns:

the destination

#### Exceptions:

*cms::CMSException* (p. 1160) if resolution failed.

*decaf::lang::exceptions::IllegalStateException* (p. 1998) if the destination resolver property is NULL.

**6.193.3.9** virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver \* *destRes*) [inline, virtual]

**6.193.3.10** virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1181).

Referenced by **activemq::cmsutil::CmsTemplate::setPubSubDomain()**.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsDestinationAccessor.h

## 6.194 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

#include <src/main/cms/CMSException.h> Inheritance diagram for cms::CMSException:

### Public Member Functions

- **CMSException** () throw ()
- **CMSException** (const **CMSException** &ex) throw ()
- **CMSException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSException** () throw ()
- virtual std::string **getMessage** () const  
*Gets the cause of the error.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual void **printStackTrace** () const  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const  
*Gets the stack trace as one contiguous string.*
- virtual const char \* **what** () const throw ()  
*Overloads the std::exception what() (p. 1162) function to return the cause of the exception.*

### 6.194.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type `std::exception` and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSEException** (p. 1160). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSEException** (p. 1160) instances.

**Since:**

1.0

## 6.194.2 Constructor & Destructor Documentation

**6.194.2.1** `cms::CMSEException::CMSEException () throw ()`

**6.194.2.2** `cms::CMSEException::CMSEException (const CMSEException & ex) throw ()`

**6.194.2.3** `cms::CMSEException::CMSEException (const std::string & message, const std::exception * cause) throw ()`

**6.194.2.4** `cms::CMSEException::CMSEException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`

**6.194.2.5** `virtual cms::CMSEException::~~CMSEException () throw () [virtual]`

## 6.194.3 Member Function Documentation

**6.194.3.1** `virtual const std::exception* cms::CMSEException::getCause () const [virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

**Returns:**

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

**6.194.3.2** `virtual std::string cms::CMSEException::getMessage () const [virtual]`

Gets the cause of the error.

**Returns:**

string errors message

**6.194.3.3** `virtual std::vector< std::pair< std::string, int> > cms::CMSEException::getStackTrace () const [virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.



**Returns:**

vector containing stack trace strings

**6.194.3.4 virtual std::string cms::CMSException::getStackTraceString () const [virtual]**

Gets the stack trace as one contiguous string.

**Returns:**

string with formatted stack trace data

**6.194.3.5 virtual void cms::CMSException::printStackTrace (std::ostream & *stream*) const [virtual]**

Prints the stack trace to the given output stream.

**Parameters:**

*stream* the target output stream.

**6.194.3.6 virtual void cms::CMSException::printStackTrace () const [virtual]**

Prints the stack trace to std::err.

**6.194.3.7 virtual void cms::CMSException::setMark (const char \* *file*, const int *lineNumber*) [virtual]**

Adds a file/line number to the stack trace.

**Parameters:**

*file* The name of the file calling this method (use `__FILE__`).

*lineNumber* The line number in the calling file (use `__LINE__`).

**6.194.3.8 virtual const char\* cms::CMSException::what () const throw () [virtual]**

Overloads the std::exception **what()** (p. 1162) function to return the cause of the exception.

**Returns:**

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

## 6.195 activemq::util::CMSExceptionSupport Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

### Public Member Functions

- virtual `~CMSExceptionSupport()`

### Static Public Member Functions

- static `cms::CMSException create (const std::string &msg, const decaf::lang::Exception &cause)`
- static `cms::CMSException create (const decaf::lang::Exception &cause)`
- static `cms::MessageEOFException createMessageEOFException (const decaf::lang::Exception &cause)`
- static `cms::MessageFormatException createMessageFormatException (const decaf::lang::Exception &cause)`

### 6.195.1 Constructor & Destructor Documentation

**6.195.1.1** virtual `activemq::util::CMSExceptionSupport::~~CMSExceptionSupport()` [virtual]

### 6.195.2 Member Function Documentation

**6.195.2.1** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & cause)` [static]

**6.195.2.2** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & msg, const decaf::lang::Exception & cause)` [static]

**6.195.2.3** static `cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException (const decaf::lang::Exception & cause)` [static]

**6.195.2.4** static `cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException (const decaf::lang::Exception & cause)` [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

## 6.196 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

#include <src/main/cms/CMSProperties.h> Inheritance diagram for cms::CMSProperties:

### Public Member Functions

- virtual **~CMSProperties** ()
- virtual bool **isEmpty** () const =0  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const =0  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const =0  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)=0  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const =0  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)=0  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const =0  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const **CMSProperties** \*source)=0  
*Copies the contents of the given properties object to this one.*
- virtual **CMSProperties** \* **clone** () const =0  
*Clones this object.*
- virtual void **clear** ()=0  
*Clears all properties from the map.*
- virtual std::string **toString** () const =0  
*Formats the contents of the Properties Object into a string that can be logged, etc.*

## 6.196.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

**Since:**

1.1

## 6.196.2 Constructor & Destructor Documentation

**6.196.2.1** virtual cms::CMSProperties::~~CMSProperties () [inline, virtual]

## 6.196.3 Member Function Documentation

**6.196.3.1** virtual void cms::CMSProperties::clear () [pure virtual]

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 479).

**6.196.3.2** virtual CMSProperties\* cms::CMSProperties::clone () const [pure virtual]

Clones this object.

**Returns:**

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 479).

**6.196.3.3** virtual void cms::CMSProperties::copy (const CMSProperties \* *source*) [pure virtual]

Copies the contents of the given properties object to this one.

**Parameters:**

*source* The source properties object.

**6.196.3.4** virtual std::string cms::CMSProperties::getProperty (const std::string & *name*, const std::string & *defaultValue*) const [pure virtual]

Looks up the value for the given property.

**Parameters:**

*name* the name of the property to be looked up.

*defaultValue* The value to be returned if the given property does not exist.

**Returns:**

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in **activemq::util::ActiveMQProperties** (p. 480).

**6.196.3.5** `virtual const char* cms::CMSProperties::getProperty (const std::string & name) const` [pure virtual]

Looks up the value for the given property.

**Parameters:**

*name* The name of the property to be looked up.

**Returns:**

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in `activemq::util::ActiveMQProperties` (p. 480).

**6.196.3.6** `virtual bool cms::CMSProperties::hasProperty (const std::string & name) const` [pure virtual]

Check to see if the Property exists in the set.

**Parameters:**

*name* the name of the property to check

**Returns:**

true if property exists, false otherwise.

Implemented in `activemq::util::ActiveMQProperties` (p. 480).

**6.196.3.7** `virtual bool cms::CMSProperties::isEmpty () const` [pure virtual]

Returns true if the properties object is empty.

**Returns:**

true if empty

Implemented in `activemq::util::ActiveMQProperties` (p. 481).

**6.196.3.8** `virtual void cms::CMSProperties::remove (const std::string & name)` [pure virtual]

Removes the property with the given name.

**Parameters:**

*name* the name of the property to be removed.s

Implemented in `activemq::util::ActiveMQProperties` (p. 481).

**6.196.3.9** virtual void cms::CMSProperties::setProperty (const std::string & *name*, const std::string & *value*) [pure virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

**Parameters:**

*name* The name of the value to be written.

*value* The value to be written.

Implemented in **activemq::util::ActiveMQProperties** (p. 481).

**6.196.3.10** virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray () const [pure virtual]

Method that serializes the contents of the property map to an array.

**Returns:**

list of pairs where the first is the name and the second is the value.

Implemented in **activemq::util::ActiveMQProperties** (p. 481).

**6.196.3.11** virtual std::string cms::CMSProperties::toString () const [pure virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

**Returns:**

string value of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 481).

The documentation for this class was generated from the following file:

- src/main/cms/CMSProperties.h

## 6.197 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

#include <src/main/cms/CMSSecurityException.h> Inheritance diagram for cms::CMSSecurityException:

### Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

### 6.197.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since:

1.3

### 6.197.2 Constructor & Destructor Documentation

**6.197.2.1** cms::CMSSecurityException::CMSSecurityException () throw ()

**6.197.2.2** cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & ex) throw ()

**6.197.2.3** cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception \* cause) throw ()

**6.197.2.4** cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()

**6.197.2.5** virtual cms::CMSSecurityException::~~CMSSecurityException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/CMSSecurityException.h



## 6.198 activemq::cmsutil::CmsTemplate Class Reference

**CmsTemplate** (p. 1170) simplifies performing synchronous CMS operations.

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate:

### Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

### Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory \*connectionFactory)
- virtual ~**CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination \*defaultDestination)  
*Sets the destination object to be used by default for send/receive operations.*
- virtual const cms::Destination \* **getDefaultDestination** () const  
*Retrieves the default destination to be used for send/receive operations.*
- virtual cms::Destination \* **getDefaultDestination** ()  
*Retrieves the default destination to be used for send/receive operations.*
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)  
*Sets the name of the default destination to be used from send/receive operations.*
- virtual const std::string **getDefaultDestinationName** () const  
*Gets the name of the default destination to be used for send/receive operations.*
- virtual void **setPubSubDomain** (bool pubSubDomain)  
*Indicates whether the default destination is a topic (true) or a queue (false).*
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)  
*Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.*

- virtual bool **isExplicitQosEnabled** () const  
*If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.*
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)  
*Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").*
- virtual void **setDeliveryMode** (int deliveryMode)  
*Set the delivery mode to use when sending a message.*
- virtual int **getDeliveryMode** () const  
*Return the delivery mode to use when sending a message.*
- virtual void **setPriority** (int priority)  
*Set the priority of a message when sending.*
- virtual int **getPriority** () const  
*Return the priority of a message when sending.*
- virtual void **setTimeToLive** (long long timeToLive)  
*Set the time-to-live of the message when sending.*
- virtual long long **getTimeToLive** () const  
*Return the time-to-live of the message when sending.*
- virtual void **execute** (SessionCallback \*action) throw ( cms::CMSEException )  
*Executes the given action within a CMS Session.*
- virtual void **execute** (ProducerCallback \*action) throw ( cms::CMSEException )  
*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (cms::Destination \*dest, ProducerCallback \*action) throw ( cms::CMSEException )  
*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (const std::string &destinationName, ProducerCallback \*action) throw ( cms::CMSEException )  
*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **send** (MessageCreator \*messageCreator) throw ( cms::CMSEException )  
*Convenience method for sending a message to the default destination.*
- virtual void **send** (cms::Destination \*dest, MessageCreator \*messageCreator) throw ( cms::CMSEException )  
*Convenience method for sending a message to the specified destination.*
- virtual void **send** (const std::string &destinationName, MessageCreator \*messageCreator) throw ( cms::CMSEException )

*Convenience method for sending a message to the specified destination.*

- virtual **cms::Message \* receive** () throw ( cms::CMSEException )  
*Performs a synchronous read from the default destination.*
- virtual **cms::Message \* receive** (cms::Destination \*destination) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message \* receive** (const std::string &destinationName) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message \* receiveSelected** (const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read consuming only messages identified by the given selector.*
- virtual **cms::Message \* receiveSelected** (cms::Destination \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*
- virtual **cms::Message \* receiveSelected** (const std::string &destinationName, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*

## Static Public Attributes

- static const long long **RECEIVE\_TIMEOUT\_NO\_WAIT**  
*Timeout value indicating that a receive operation should check if a message is immediately available without blocking.*
- static const long long **RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT**  
*Timeout value indicating a blocking receive without timeout.*
- static const int **DEFAULT\_PRIORITY**  
*Default message priority.*
- static const long long **DEFAULT\_TIME\_TO\_LIVE**  
*My default, messages should live forever.*

## Protected Member Functions

- **CmsTemplate** (const **CmsTemplate** &)
- **CmsTemplate & operator=** (const **CmsTemplate** &)
- void **init** () throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )  
*Initializes this object and prepares it for use.*

- void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )

*Clears all internal resources.*

## Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

### 6.198.1 Detailed Description

**CmsTemplate** (p. 1170) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's **JmsTemplate** is for JMS. Provided with a **CMS ConnectionFactory**, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 1170) the user must first set the destination (either by name or by setting the destination object directly) and then call **init** to initialize the object for use.

**CmsTemplate** (p. 1170) allows the user to get access to a **CMS Session** through a user-defined **SessionCallback** (p. 3378). Similarly, if the user wants direct access to a **CMS MessageProducer**, it can provide a **ProducerCallback** (p. 3064). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the **send** methods.

See also:

**SessionCallback** (p. 3378)  
**ProducerCallback** (p. 3064)  
**MessageCreator** (p. 2593)

## 6.198.2 Constructor & Destructor Documentation

**6.198.2.1** `activemq::cmsutil::CmsTemplate::CmsTemplate (const CmsTemplate &)`  
[inline, protected]

**6.198.2.2** `activemq::cmsutil::CmsTemplate::CmsTemplate ()`

**6.198.2.3** `activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * connectionFactory)`

**6.198.2.4** `virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate ()` [virtual]

## 6.198.3 Member Function Documentation

**6.198.3.1** `void activemq::cmsutil::CmsTemplate::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`  
[protected, virtual]

Clears all internal resources.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1158).

**6.198.3.2** `virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & destinationName, ProducerCallback * action) throw (cms::CMSException)` [virtual]

Executes the given action and provides it with a CMS Session and producer.

### Parameters:

*destinationName* the name of the destination to send messages to (to internally be resolved to an actual destination)

*action* the action to perform

### Exceptions:

*cms::CMSException* (p. 1160) thrown if an error occurs.

**6.198.3.3** `virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * dest, ProducerCallback * action) throw (cms::CMSException)`  
[virtual]

Executes the given action and provides it with a CMS Session and producer.

### Parameters:

*dest* the destination to send messages to

*action* the action to perform

### Exceptions:

*cms::CMSException* (p. 1160) thrown if an error occurs.

#### 6.198.3.4 **virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback \* *action*) throw ( cms::CMSException ) [virtual]**

Executes the given action and provides it with a CMS Session and producer.

##### Parameters:

*action* the action to perform

##### Exceptions:

*cms::CMSException* (p. 1160) thrown if an error occurs.

#### 6.198.3.5 **virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback \* *action*) throw ( cms::CMSException ) [virtual]**

Executes the given action within a CMS Session.

##### Parameters:

*action* the action to perform within a CMS Session

##### Exceptions:

*cms::CMSException* (p. 1160) thrown if an error occurs.

#### 6.198.3.6 **virtual cms::Destination\* activemq::cmsutil::CmsTemplate::getDefaultDestination () [inline, virtual]**

Retrieves the default destination to be used for send/receive operations.

##### Returns:

the default destination. Non-const version of this method.

#### 6.198.3.7 **virtual const cms::Destination\* activemq::cmsutil::CmsTemplate::getDefaultDestination () const [inline, virtual]**

Retrieves the default destination to be used for send/receive operations.

##### Returns:

the default destination. Const version of this method.

#### 6.198.3.8 **virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const [inline, virtual]**

Gets the name of the default destination to be used for send/receive operations. The destination type (topic/queue) is determined by the `pubSubDomain` property.

**Returns:**

the default name of the destination for send/receive operations.

**6.198.3.9** `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const`  
[inline, virtual]

Return the delivery mode to use when sending a message.

**6.198.3.10** `virtual int activemq::cmsutil::CmsTemplate::getPriority () const`  
[inline, virtual]

Return the priority of a message when sending.

**6.198.3.11** `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const` [inline, virtual]

**6.198.3.12** `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const` [inline, virtual]

Return the time-to-live of the message when sending.

**6.198.3.13** `void activemq::cmsutil::CmsTemplate::init () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
[protected, virtual]

Initializes this object and prepares it for use. This should be called before any other methods are called.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1158).

**6.198.3.14** `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled () const` [inline, virtual]

If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message. Otherwise, the default values, that may be set administratively, will be used.

**Returns:**

true if overriding default values of QOS parameters (deliveryMode, priority, and timeToLive)

**See also:**

`setDeliveryMode` (p. 1180)  
`setPriority` (p. 1181)  
`setTimeToLive` (p. 1181)

**6.198.3.15** `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled () const [inline, virtual]`

**6.198.3.16** `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled () const [inline, virtual]`

**6.198.3.17** `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal () const [inline, virtual]`

**6.198.3.18** `CmsTemplate& activemq::cmsutil::CmsTemplate::operator= (const CmsTemplate &) [inline, protected]`

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1159).

**6.198.3.19** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & destinationName) throw ( cms::CMSException ) [virtual]`

Performs a synchronous read from the specified destination.

**Parameters:**

*destinationName* the name of the destination to receive on (will be resolved to destination internally).

**Returns:**

the message

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs

**6.198.3.20** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * destination) throw ( cms::CMSException ) [virtual]`

Performs a synchronous read from the specified destination.

**Parameters:**

*destination* the destination to receive on

**Returns:**

the message

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs



**6.198.3.21**    `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive () throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the default destination.

**Returns:**

the message

**Exceptions:**

*cms::CMSEException* (p. 1160) thrown if an error occurs

**6.198.3.22**    `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & destinationName, const std::string & selector) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

**Parameters:**

*destinationName* the name of the destination to receive on (will be resolved to destination internally).

*selector* the selector expression.

**Returns:**

the message

**Exceptions:**

*cms::CMSEException* (p. 1160) thrown if an error occurs

**6.198.3.23**    `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (cms::Destination * destination, const std::string & selector) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

**Parameters:**

*destination* the destination to receive on.

*selector* the selector expression.

**Returns:**

the message

**Exceptions:**

*cms::CMSEException* (p. 1160) thrown if an error occurs

**6.198.3.24** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & selector) throw ( cms::CMSException ) [virtual]`

Performs a synchronous read consuming only messages identified by the given selector.

**Parameters:**

*selector* the selector expression.

**Returns:**

the message

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs

**6.198.3.25** `virtual void activemq::cmsutil::CmsTemplate::send (const std::string & destinationName, MessageCreator * messageCreator) throw ( cms::CMSException ) [virtual]`

Convenience method for sending a message to the specified destination.

**Parameters:**

*destinationName* The name of the destination to send to.

*messageCreator* Responsible for creating the message to be sent

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs.

**6.198.3.26** `virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * dest, MessageCreator * messageCreator) throw ( cms::CMSException ) [virtual]`

Convenience method for sending a message to the specified destination.

**Parameters:**

*dest* The destination to send to

*messageCreator* Responsible for creating the message to be sent

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs.

**6.198.3.27** `virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * messageCreator) throw ( cms::CMSException ) [virtual]`

Convenience method for sending a message to the default destination.

**Parameters:**

*messageCreator* Responsible for creating the message to be sent

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs.

**6.198.3.28 virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination \* *defaultDestination*) [inline, virtual]**

Sets the destination object to be used by default for send/receive operations. If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

**Parameters:**

*defaultDestination* the default destination

**6.198.3.29 virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & *defaultDestinationName*) [inline, virtual]**

Sets the name of the default destination to be used from send/receive operations. Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

**Parameters:**

*defaultDestinationName* the name of the destination for send/receive to by default.

**6.198.3.30 virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int *deliveryMode*) [inline, virtual]**

Set the delivery mode to use when sending a message. Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

**Parameters:**

*deliveryMode* the delivery mode to use

**See also:**

`isExplicitQosEnabled` (p. 1176)

**6.198.3.31**    **virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent**  
(bool *deliveryPersistent*)    [inline, virtual]

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false"). This will set the delivery mode accordingly, to either "PERSISTENT" or "NON\_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also:

**setDeliveryMode(int)** (p. 1180)

**6.198.3.32**    **virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled**  
(bool *explicitQosEnabled*)    [inline, virtual]

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also:

**setDeliveryMode** (p. 1180)

**setPriority** (p. 1181)

**setTimeToLive** (p. 1181)

**6.198.3.33**    **virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled**  
(bool *messageIdEnabled*)    [inline, virtual]

**6.198.3.34**    **virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled** (bool  
*messageTimestampEnabled*)    [inline, virtual]

**6.198.3.35**    **virtual void activemq::cmsutil::CmsTemplate::setNoLocal** (bool *noLocal*)  
[inline, virtual]

**6.198.3.36**    **virtual void activemq::cmsutil::CmsTemplate::setPriority** (int *priority*)  
[inline, virtual]

Set the priority of a message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also:

**isExplicitQosEnabled** (p. 1176)

**6.198.3.37**    **virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain** (bool  
*pubSubDomain*)    [inline, virtual]

Indicates whether the default destination is a topic (true) or a queue (false). Calling this method will set the defaultDestination property to NULL.

Parameters:

*pubSubDomain* indicates whether to use pub-sub messaging (topics).

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1159).

References **activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()**.

**6.198.3.38**    **virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout** (long long *receiveTimeout*) [inline, virtual]

**6.198.3.39**    **virtual void activemq::cmsutil::CmsTemplate::setTimeToLive** (long long *timeToLive*) [inline, virtual]

Set the time-to-live of the message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

**Parameters:**

*timeToLive* the message's lifetime (in milliseconds)

See also:

**isExplicitQosEnabled** (p. 1176)

## 6.198.4 Friends And Related Function Documentation

**6.198.4.1**    friend class **ProducerExecutor** [friend]

**6.198.4.2**    friend class **ReceiveExecutor** [friend]

**6.198.4.3**    friend class **ResolveProducerExecutor** [friend]

**6.198.4.4**    friend class **ResolveReceiveExecutor** [friend]

**6.198.4.5**    friend class **SendExecutor** [friend]

## 6.198.5 Field Documentation

**6.198.5.1**    **const int activemq::cmsutil::CmsTemplate::DEFAULT\_PRIORITY**  
[static]

Default message priority.

**6.198.5.2**    **const long long activemq::cmsutil::CmsTemplate::DEFAULT\_TIME\_TO\_LIVE** [static]

My default, messages should live forever.

**6.198.5.3**    **const long long activemq::cmsutil::CmsTemplate::RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT** [static]

Timeout value indicating a blocking receive without timeout.

#### 6.198.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_ - TIMEOUT_NO_WAIT [static]`

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.199 code Struct Reference

```
#include <src/main/decaf/internal/util/zip/inftrees.h>
```

### Data Fields

- unsigned char **op**
- unsigned char **bits**
- unsigned short **val**

### 6.199.1 Field Documentation

**6.199.1.1 unsigned char code::bits**

**6.199.1.2 unsigned char code::op**

**6.199.1.3 unsigned short code::val**

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/inftrees.h`

## 6.200 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>Inheritance          diagram          for
decaf::util::Collection< E >:
```

### Public Member Functions

- virtual **~Collection** ()
- virtual bool **add** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Returns true if this collection changed as a result of the call.*
- virtual bool **addAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection.*
- virtual void **clear** ()=0 throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).*
- virtual bool **contains** (const E &value) const =0 throw ( lang::Exception )  
*Returns true if this collection contains the specified element.*
- virtual bool **containsAll** (const **Collection**< E > &collection) const =0 throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **equals** (const **Collection**< E > &value) const =0  
*Compares the passed collection to this one, if they contain the same elements, i.e.*
- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from the collection.*
- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes all this collection's elements that are also contained in the specified collection (optional operation).*
- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Retains only the elements in this collection that are contained in the specified collection (optional operation).*



- virtual std::size\_t **size** () const =0

*Returns the number of elements in this collection.*

- virtual std::vector< E > **toArray** () const =0

*Returns an array containing all of the elements in this collection.*

### 6.200.1 Detailed Description

**template<typename E> class decaf::util::Collection< E >**

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 1184) implementation classes (which typically implement **Collection** (p. 1184) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 1184), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 1184) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw `UnsupportedOperationException` if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an `UnsupportedOperationException` if the invocation would have no effect on the collection. For example, invoking the `addAll(Collection)` method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the `equals` method. For example, the specification for the `contains(Object o)` method says: returns true if and only if this collection contains at least one element `e` such that `(o==null ? e==null : o.equals(e))`.

**Since:**

1.0

## 6.200.2 Constructor & Destructor Documentation

**6.200.2.1** `template<typename E> virtual decaf::util::Collection< E >::~~Collection  
( ) [inline, virtual]`

## 6.200.3 Member Function Documentation

**6.200.3.1** `template<typename E> virtual bool decaf::util::Collection<  
E >::add (const E & value) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException,  
lang::exceptions::IllegalStateException ) [pure  
virtual]`

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1184) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

### Parameters:

*value* - reference to the element to add.

### Returns:

true if the element was added

### Exceptions:

*UnsupportedOperationException*

*IllegalArgumentException*

*IllegalStateException* if the element cannot be added at this time due to insertion restrictions

Implemented in `decaf::util::AbstractQueue< E >` (p.195), `decaf::util::PriorityQueue< E >` (p.3031), `decaf::util::StlList< E >` (p.3594), `decaf::util::StlSet< E >` (p.3625), `decaf::util::StlList< cms::MessageConsumer * >` (p.3594), `decaf::util::StlList< CompositeTask * >` (p.3594), `decaf::util::StlList< URI >` (p.3594), `decaf::util::StlList< Pointer< DestinationInfo > >` (p.3594), `decaf::util::StlList< PrimitiveValueNode >` (p.3594), `decaf::util::StlList< Pointer< Command > >` (p.3594), `decaf::util::StlList< Pointer< BackupTransport > >` (p.3594), `decaf::util::StlList< cms::MessageProducer * >` (p.3594), `decaf::util::StlList< cms::Destination * >` (p.3594), `decaf::util::StlList< cms::Session * >` (p.3594), `decaf::util::StlList< cms::Connection * >` (p.3594), `decaf::util::StlSet< transport::TransportListener * >` (p.3625), `decaf::util::StlSet< Pointer< Synchronization > >` (p.3625), `decaf::util::StlSet< Resource * >` (p.3625), and `decaf::util::StlSet< ActiveMQSession * >` (p.3625).

```

6.200.3.2  template<typename E> virtual bool decaf::util::Collection<
           E >::addAll (const Collection< E > & collection) throw
           ( lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException,
           lang::exceptions::IllegalStateException ) [pure
           virtual]

```

Adds all of the elements in the specified collection to this collection. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

**Parameters:**

*collection* - **Collection** (p. 1184) whose elements are added to this one.

**Returns:**

true if this collection changed as a result of the call

**Exceptions:**

*UnsupportedOperationException*

*IllegalArgumentException*

*IllegalStateException* if the element cannot be added at this time due to insertion restrictions

```

6.200.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear
           () throw ( lang::exceptions::UnsupportedOperationException ) [pure
           virtual]

```

Removes all of the elements from this collection (optional operation). This collection will be empty after this method returns unless it throws an exception.

**Exceptions:**

*UnsupportedOperationException*

Implemented in **decaf::util::AbstractCollection< E >** (p. 183), **decaf::util::AbstractQueue< E >** (p. 196), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3718), **decaf::util::PriorityQueue< E >** (p. 3031), **decaf::util::StlList< E >** (p. 3596), **decaf::util::StlSet< E >** (p. 3626), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 183), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 183), **decaf::util::AbstractCollection< Resource \* >** (p. 183), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 183), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 183), **decaf::util::AbstractCollection< URI >** (p. 183), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 183), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 183), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 183), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 183), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 183), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 183), **decaf::util::AbstractCollection< cms::Destination \* >**

(p. 183), `decaf::util::AbstractCollection< cms::Session * >` (p. 183), `decaf::util::AbstractCollection< cms::Connection * >` (p. 183), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3596), `decaf::util::StlList< CompositeTask * >` (p. 3596), `decaf::util::StlList< URI >` (p. 3596), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3596), `decaf::util::StlList< PrimitiveValueNode >` (p. 3596), `decaf::util::StlList< Pointer< Command > >` (p. 3596), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3596), `decaf::util::StlList< cms::MessageProducer * >` (p. 3596), `decaf::util::StlList< cms::Destination * >` (p. 3596), `decaf::util::StlList< cms::Session * >` (p. 3596), `decaf::util::StlList< cms::Connection * >` (p. 3596), `decaf::util::StlSet< transport::TransportListener * >` (p. 3626), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3626), `decaf::util::StlSet< Resource * >` (p. 3626), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3626).

**6.200.3.4** `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value) const throw ( lang::Exception ) [pure virtual]`

Returns true if this collection contains the specified element. More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)).

**Parameters:**

*value* - value to check for presence in the collection

**Returns:**

true if there is at least one of the elements in the collection

**Exceptions:**

*Exception*

Implemented in `decaf::util::AbstractCollection< E >` (p. 184), `decaf::util::StlList< E >` (p. 3596), `decaf::util::StlSet< E >` (p. 3627), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 184), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 184), `decaf::util::AbstractCollection< Resource * >` (p. 184), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 184), `decaf::util::AbstractCollection< CompositeTask * >` (p. 184), `decaf::util::AbstractCollection< URI >` (p. 184), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 184), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 184), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 184), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 184), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 184), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 184), `decaf::util::AbstractCollection< cms::Destination * >` (p. 184), `decaf::util::AbstractCollection< cms::Session * >` (p. 184), `decaf::util::AbstractCollection< cms::Connection * >` (p. 184), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3596), `decaf::util::StlList< CompositeTask * >` (p. 3596), `decaf::util::StlList< URI >` (p. 3596), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3596), `decaf::util::StlList< PrimitiveValueNode >` (p. 3596), `decaf::util::StlList< Pointer< Command > >` (p. 3596), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3596), `decaf::util::StlList< cms::MessageProducer * >` (p. 3596), `decaf::util::StlList< cms::Destination * >` (p. 3596), `decaf::util::StlList< cms::Session * >` (p. 3596), `decaf::util::StlList< cms::Connection * >` (p. 3596), `decaf::util::StlSet< transport::TransportListener * >` (p. 3627), `decaf::util::StlSet<`

**Pointer< Synchronization > >** (p. 3627), **decaf::util::StlSet< Resource \* >** (p. 3627), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3627).

**6.200.3.5** `template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception ) [pure virtual]`

Returns true if this collection contains all of the elements in the specified collection.

**Parameters:**

*collection* - **Collection** (p. 1184) to compare to this one.

**Exceptions:**

*Exception*

**6.200.3.6** `template<typename E> virtual bool decaf::util::Collection< E >::equals (const Collection< E > & value) const [pure virtual]`

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

**Returns:**

true if the Collections contain the same elements.

Implemented in **decaf::util::StlList< E >** (p. 3597), and **decaf::util::StlSet< E >** (p. 3627).

**6.200.3.7** `template<typename E> virtual bool decaf::util::Collection< E >::isEmpty () const [pure virtual]`

**Returns:**

true if this collection contains no elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 186), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3721), **decaf::util::StlList< E >** (p. 3598), **decaf::util::StlSet< E >** (p. 3627), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 186), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 186), **decaf::util::AbstractCollection< Resource \* >** (p. 186), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 186), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 186), **decaf::util::AbstractCollection< URI >** (p. 186), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 186), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 186), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 186), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 186), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 186), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 186), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 186), **decaf::util::AbstractCollection< cms::Session \* >** (p. 186), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 186), **decaf::util::StlList<**

`cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), `decaf::util::StlList< cms::Connection * >` (p. 3598), `decaf::util::StlSet< transport::TransportListener * >` (p. 3627), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3627), `decaf::util::StlSet< Resource * >` (p. 3627), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3627).

**6.200.3.8** `template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes a single instance of the specified element from the collection. More formally, removes an element *e* such that (`o==null ? e==null : o.equals(e)`), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

#### Parameters:

*value* - reference to the element to remove.

#### Returns:

true if the collection was changed

#### Exceptions:

*UnsupportedOperationException*

*IllegalArgumentException*

Implemented in `decaf::util::AbstractCollection< E >` (p. 187), `decaf::util::PriorityQueue< E >` (p. 3034), `decaf::util::StlList< E >` (p. 3600), `decaf::util::StlSet< E >` (p. 3628), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 187), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 187), `decaf::util::AbstractCollection< Resource * >` (p. 187), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 187), `decaf::util::AbstractCollection< CompositeTask * >` (p. 187), `decaf::util::AbstractCollection< URI >` (p. 187), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 187), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 187), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 187), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 187), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 187), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 187), `decaf::util::AbstractCollection< cms::Destination * >` (p. 187), `decaf::util::AbstractCollection< cms::Session * >` (p. 187), `decaf::util::AbstractCollection< cms::Connection * >` (p. 187), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3600), `decaf::util::StlList< CompositeTask * >` (p. 3600), `decaf::util::StlList< URI >` (p. 3600), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3600), `decaf::util::StlList< PrimitiveValueNode >` (p. 3600), `decaf::util::StlList< Pointer< Command > >` (p. 3600), `decaf::util::StlList< Pointer<`

BackupTransport > > (p. 3600), decaf::util::StlList< cms::MessageProducer \* > (p. 3600), decaf::util::StlList< cms::Destination \* > (p. 3600), decaf::util::StlList< cms::Session \* > (p. 3600), decaf::util::StlList< cms::Connection \* > (p. 3600), decaf::util::StlSet< transport::TransportListener \* > (p. 3628), decaf::util::StlSet< Pointer< Synchronization > > (p. 3628), decaf::util::StlSet< Resource \* > (p. 3628), and decaf::util::StlSet< ActiveMQSession \* > (p. 3628).

**6.200.3.9**    template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & *collection*)  
               throw ( lang::exceptions::UnsupportedOperationException,  
               lang::exceptions::IllegalArgumentException ) [pure virtual]

Removes all this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

**Parameters:**

*collection* - The **Collection** (p. 1184) whose elements are to be removed

**Returns:**

true if the collection changed as a result of this call

**Exceptions:**

*UnsupportedOperationException*

*IllegalArgumentException*

**6.200.3.10**    template<typename E> virtual bool decaf::util::Collection< E >::retainAll (const Collection< E > & *collection*)  
               throw ( lang::exceptions::UnsupportedOperationException,  
               lang::exceptions::IllegalArgumentException ) [pure virtual]

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

**Parameters:**

*collection* - The **Collection** (p. 1184) whose elements are to be retained

**Returns:**

true if the collection changed as a result of this call

**Exceptions:**

*UnsupportedOperationException*

*IllegalArgumentException*

**6.200.3.11** `template<typename E> virtual std::size_t decaf::util::Collection< E >::size () const [pure virtual]`

Returns the number of elements in this collection. If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**Returns:**

the number of elements in this collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3724), `decaf::util::PriorityQueue< E >` (p. 3034), `decaf::util::StlList< E >` (p. 3601), `decaf::util::StlSet< E >` (p. 3628), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3601), `decaf::util::StlList< CompositeTask * >` (p. 3601), `decaf::util::StlList< URI >` (p. 3601), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3601), `decaf::util::StlList< PrimitiveValueNode >` (p. 3601), `decaf::util::StlList< Pointer< Command > >` (p. 3601), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3601), `decaf::util::StlList< cms::MessageProducer * >` (p. 3601), `decaf::util::StlList< cms::Destination * >` (p. 3601), `decaf::util::StlList< cms::Session * >` (p. 3601), `decaf::util::StlList< cms::Connection * >` (p. 3601), `decaf::util::StlSet< transport::TransportListener * >` (p. 3628), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3628), `decaf::util::StlSet< Resource * >` (p. 3628), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3628).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::equals()`, `decaf::util::AbstractCollection< cms::Connection * >::isEmpty()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::toArray()`.

**6.200.3.12** `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]`

Returns an array containing all of the elements in this collection. If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

**Returns:**

an array of the elements in this collection.

Implemented in `decaf::util::AbstractCollection< E >` (p. 189), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3724), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 189), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 189), `decaf::util::AbstractCollection< Resource * >` (p. 189), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 189), `decaf::util::AbstractCollection< CompositeTask * >` (p. 189), `decaf::util::AbstractCollection< URI >` (p. 189), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 189), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 189), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 189), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 189), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 189), and `decaf::util::AbstractCollection< cms::MessageProducer`



\* > (p. 189), `decaf::util::AbstractCollection< cms::Destination * >` (p. 189), `decaf::util::AbstractCollection< cms::Session * >` (p. 189), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 189).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

## 6.201 activemq::commands::Command Class Reference

#include <src/main/activemq/commands/Command.h> Inheritance diagram for activemq::commands::Command:

### Public Member Functions

- virtual **~Command** ()
- virtual void **setCommandId** (int id)=0  
*Sets the **Command** (p. 1194) Id of this **Message** (p. 2516).*
- virtual int **getCommandId** () const =0  
*Gets the **Command** (p. 1194) Id of this **Message** (p. 2516).*
- virtual void **setResponseRequired** (const bool required)=0  
*Set if this **Message** (p. 2516) requires a **Response** (p. 3285).*
- virtual bool **isResponseRequired** () const =0  
*Is a **Response** (p. 3285) required for this **Command** (p. 1194).*
- virtual std::string **toString** () const =0  
*Returns a provider-specific string that provides information about the contents of the command.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** \*visitor)=0 throw ( **exceptions::ActiveMQException** )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

## 6.201.1 Constructor & Destructor Documentation

**6.201.1.1** virtual `activemq::commands::Command::~~Command ()` [inline, virtual]

## 6.201.2 Member Function Documentation

**6.201.2.1** virtual `int activemq::commands::Command::getCommandId ()` const [pure virtual]

Gets the **Command** (p. 1194) Id of this **Message** (p. 2516).

### Returns:

**Command** (p. 1194) Id

Implemented in `activemq::commands::BaseCommand` (p. 760).

**6.201.2.2** virtual `bool activemq::commands::Command::isBrokerInfo ()` const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 760), and `activemq::commands::BrokerInfo` (p. 900).

**6.201.2.3** virtual `bool activemq::commands::Command::isConnectionInfo ()` const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::ConnectionInfo` (p. 1358).

**6.201.2.4** virtual `bool activemq::commands::Command::isConsumerInfo ()` const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::ConsumerInfo` (p. 1463).

**6.201.2.5** virtual `bool activemq::commands::Command::isKeepAliveInfo ()` const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::KeepAliveInfo` (p. 2268).

**6.201.2.6** virtual `bool activemq::commands::Command::isMessage ()` const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::Message` (p. 2527).

**6.201.2.7** `virtual bool activemq::commands::Command::isMessageAck () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::MessageAck` (p. 2562).

**6.201.2.8** `virtual bool activemq::commands::Command::isMessageDispatch () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::MessageDispatch` (p. 2596).

**6.201.2.9** `virtual bool activemq::commands::Command::isMessageDispatchNotification () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 761), and `activemq::commands::MessageDispatchNotification` (p. 2632).

**6.201.2.10** `virtual bool activemq::commands::Command::isProducerAck () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 762), and `activemq::commands::ProducerAck` (p. 3038).

**6.201.2.11** `virtual bool activemq::commands::Command::isProducerInfo () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 762), and `activemq::commands::ProducerInfo` (p. 3098).

**6.201.2.12** `virtual bool activemq::commands::Command::isRemoveInfo () const`  
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 762), and `activemq::commands::RemoveInfo` (p. 3196).

**6.201.2.13** `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 762), and `activemq::commands::RemoveSubscriptionInfo` (p. 3224).

**6.201.2.14** `virtual bool activemq::commands::Command::isResponse () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 762), and `activemq::commands::Response` (p. 3287).

**6.201.2.15** `virtual bool activemq::commands::Command::isResponseRequired () const [pure virtual]`

Is a **Response** (p. 3285) required for this **Command** (p. 1194).

**Returns:**

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 762).

**6.201.2.16** `virtual bool activemq::commands::Command::isShutdownInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 762), and **activemq::commands::ShutdownInfo** (p. 3472).

**6.201.2.17** `virtual bool activemq::commands::Command::isTransactionInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 763), and **activemq::commands::TransactionInfo** (p. 3849).

**6.201.2.18** `virtual bool activemq::commands::Command::isWireFormatInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 763), and **activemq::commands::WireFormatInfo** (p. 3989).

**6.201.2.19** `virtual void activemq::commands::Command::setCommandId (int id) [pure virtual]`

Sets the **Command** (p. 1194) Id of this **Message** (p. 2516).

**Parameters:**

*id* **Command** (p. 1194) Id

Implemented in **activemq::commands::BaseCommand** (p. 763).

**6.201.2.20** `virtual void activemq::commands::Command::setResponseRequired (const bool required) [pure virtual]`

Set if this **Message** (p. 2516) requires a **Response** (p. 3285).

**Parameters:**

*required* true if response is required

Implemented in **activemq::commands::BaseCommand** (p. 763).

### 6.201.2.21 virtual std::string activemq::commands::Command::toString () const [pure virtual]

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 833).

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 208), `activemq::commands::ActiveMQBytesMessage` (p. 244), `activemq::commands::ActiveMQMapMessage` (p. 372), `activemq::commands::ActiveMQMessage` (p. 400), `activemq::commands::ActiveMQObjectMessage` (p. 445), `activemq::commands::ActiveMQStreamMessage` (p. 547), `activemq::commands::ActiveMQTextMessage` (p. 666), `activemq::commands::BaseCommand` (p. 763), `activemq::commands::BrokerInfo` (p. 901), `activemq::commands::ConnectionControl` (p. 1270), `activemq::commands::ConnectionError` (p. 1298), `activemq::commands::ConnectionInfo` (p. 1359), `activemq::commands::ConsumerControl` (p. 1404), `activemq::commands::ConsumerInfo` (p. 1464), `activemq::commands::ControlCommand` (p. 1495), `activemq::commands::DataArrayResponse` (p. 1530), `activemq::commands::DataResponse` (p. 1585), `activemq::commands::DestinationInfo` (p. 1730), `activemq::commands::ExceptionResponse` (p. 1841), `activemq::commands::FlushCommand` (p. 1939), `activemq::commands::IntegerResponse` (p. 2093), `activemq::commands::KeepAliveInfo` (p. 2268), `activemq::commands::Message` (p. 2530), `activemq::commands::MessageAck` (p. 2563), `activemq::commands::MessageDispatch` (p. 2597), `activemq::commands::MessageDispatchNotification` (p. 2633), `activemq::commands::MessagePull` (p. 2742), `activemq::commands::ProducerAck` (p. 3038), `activemq::commands::ProducerInfo` (p. 3099), `activemq::commands::RemoveInfo` (p. 3196), `activemq::commands::RemoveSubscriptionInfo` (p. 3225), `activemq::commands::ReplayCommand` (p. 3253), `activemq::commands::Response` (p. 3287), `activemq::commands::SessionInfo` (p. 3409), `activemq::commands::ShutdownInfo` (p. 3472), `activemq::commands::TransactionInfo` (p. 3849), and `activemq::commands::WireFormatInfo` (p. 3991).

### 6.201.2.22 virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor \* visitor) throw ( exceptions::ActiveMQException ) [pure virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

#### Returns:

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implemented in `activemq::commands::BrokerError` (p. 866), `activemq::commands::BrokerInfo` (p. 902), `activemq::commands::ConnectionControl` (p. 1271), `activemq::commands::ConnectionError` (p. 1298), `activemq::commands::ConnectionInfo` (p. 1359), `activemq::commands::ConsumerControl` (p. 1405), `activemq::commands::ConsumerInfo` (p. 1465), `activemq::commands::ControlCommand` (p. 1495), `activemq::commands::DestinationInfo` (p. 1730), `activemq::commands::FlushCommand` (p. 1939), `activemq::commands::KeepAliveInfo` (p. 2268), `activemq::commands::Message` (p. 2531), `activemq::commands::MessageAck` (p. 2563), `activemq::commands::MessageDispatch`

(p. 2597), **activemq::commands::MessageDispatchNotification**  
(p. 2633), **activemq::commands::MessagePull** (p. 2742), **ac-**  
**tivemq::commands::ProducerAck** (p. 3038), **activemq::commands::ProducerInfo**  
(p. 3099), **activemq::commands::RemoveInfo** (p. 3196), **ac-**  
**tivemq::commands::RemoveSubscriptionInfo** (p. 3225), **ac-**  
**tivemq::commands::ReplayCommand** (p. 3253), **activemq::commands::Response**  
(p. 3287), **activemq::commands::SessionInfo** (p. 3409), **ac-**  
**tivemq::commands::ShutdownInfo** (p. 3472), **activemq::commands::TransactionInfo**  
(p. 3850), and **activemq::commands::WireFormatInfo** (p. 3991).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Command.h`

## 6.202 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

#include <src/main/activemq/state/CommandVisitor.h> Inheritance diagram for activemq::state::CommandVisitor:

### Public Member Functions

- virtual `~CommandVisitor ()`
- virtual `decaf::lang::Pointer< commands::Command > processTransactionInfo (commands::TransactionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveInfo (commands::RemoveInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processConnectionInfo (commands::ConnectionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processSessionInfo (commands::SessionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processProducerInfo (commands::ProducerInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processConsumerInfo (commands::ConsumerInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processMessage (commands::Message *send)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processMessageAck (commands::MessageAck *ack)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processMessagePull (commands::MessagePull *pull)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processBeginTransaction (commands::TransactionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processPrepareTransaction (commands::TransactionInfo *info)=0` throw ( exceptions::ActiveMQException )



- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** \*notification)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** \*ack)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** \*dispatch)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** \*control)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** \*control)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** \*replay)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** \*response)=0 throw ( exceptions::ActiveMQException )

### 6.202.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a **visit** method that is called with an instance of this interface and each one then call the appropriate **processXXX** method.

Since:

3.0

## 6.202.2 Constructor & Destructor Documentation

**6.202.2.1** `virtual activemq::state::CommandVisitor::~~CommandVisitor ()` [inline, virtual]

## 6.202.3 Member Function Documentation

**6.202.3.1** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processBeginTransaction  
(commands::TransactionInfo * info) throw (  
exceptions::ActiveMQException )` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1394).

**6.202.3.2** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processBrokerError  
(commands::BrokerError * error) throw ( exceptions::ActiveMQException  
)` [pure virtual]

**6.202.3.3** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processBrokerInfo  
(commands::BrokerInfo * info) throw ( exceptions::ActiveMQException )` [pure virtual]

**6.202.3.4** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processCommitTransactionOnePhase  
(commands::TransactionInfo * info) throw (  
exceptions::ActiveMQException )` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1394).

**6.202.3.5** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processCommitTransactionTwoPhase  
(commands::TransactionInfo * info) throw (  
exceptions::ActiveMQException )` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.6** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConnectionControl  
(commands::ConnectionControl \* *control*) throw (  
exceptions::ActiveMQException ) [pure virtual]
- 6.202.3.7** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConnectionError  
(commands::ConnectionError \* *error*) throw (  
exceptions::ActiveMQException ) [pure virtual]
- 6.202.3.8** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConnectionInfo  
(commands::ConnectionInfo \* *info*) throw (  
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.9** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConsumerControl  
(commands::ConsumerControl \* *control*) throw (  
exceptions::ActiveMQException ) [pure virtual]
- 6.202.3.10** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConsumerInfo  
(commands::ConsumerInfo \* *info*) throw (  
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.11** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processControlCommand  
(commands::ControlCommand \* *command*) throw (  
exceptions::ActiveMQException ) [pure virtual]
- 6.202.3.12** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processDestinationInfo  
(commands::DestinationInfo \* *info*) throw (  
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.13** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processEndTransaction  
(commands::TransactionInfo \* *info*) throw (  
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.14 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processFlushCommand  
(commands::FlushCommand * command) throw (  
exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.15 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processForgetTransaction  
(commands::TransactionInfo * info) throw (  
exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.16 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processKeepAliveInfo  
(commands::KeepAliveInfo * info) throw (  
exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.17 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessage (commands::Message  
* send) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.18 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageAck  
(commands::MessageAck * ack) throw ( exceptions::ActiveMQException  
) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1395).

- 6.202.3.19 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatch  
(commands::MessageDispatch * dispatch) throw (  
exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.20 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatchNotification  
(commands::MessageDispatchNotification * notification) throw (  
exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.21 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessagePull  
(commands::MessagePull * pull) throw ( exceptions::ActiveMQException  
) [pure virtual]`
- 6.202.3.22 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processPrepareTransaction  
(commands::TransactionInfo * info) throw (  
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.23** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerAck  
(commands::ProducerAck \* *ack*) throw ( exceptions::ActiveMQException  
) [pure virtual]

**6.202.3.24** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerInfo (com-  
mands::ProducerInfo \* *info*) throw ( exceptions::ActiveMQException )  
[pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.25** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRecoverTransactions  
(commands::TransactionInfo \* *info*) throw ( exceptions::ActiveMQException ) [pure virtual]

**6.202.3.26** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConnection  
(commands::ConnectionId \* *id*) throw ( exceptions::ActiveMQException  
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.27** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConsumer  
(commands::ConsumerId \* *id*) throw ( exceptions::ActiveMQException  
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.28** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveDestination  
(commands::DestinationInfo \* *info*) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.29** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveInfo  
(commands::RemoveInfo \* *info*) throw ( exceptions::ActiveMQException  
) [pure virtual]

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1211).

**6.202.3.30** virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveProducer  
(commands::ProducerId \* *id*) throw ( exceptions::ActiveMQException )  
[pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.31** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSession  
(commands::SessionId * id) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1396).

**6.202.3.32** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSubscriptionInfo  
(commands::RemoveSubscriptionInfo * info) throw (   
exceptions::ActiveMQException ) [pure virtual]`

**6.202.3.33** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processReplayCommand  
(commands::ReplayCommand * replay) throw (   
exceptions::ActiveMQException ) [pure virtual]`

**6.202.3.34** `virtual decaf::lang::Pointer<commands::Command> ac-  
tivemq::state::CommandVisitor::processResponse (commands::Response  
* response) throw ( exceptions::ActiveMQException ) [pure virtual]`

**6.202.3.35** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRollbackTransaction  
(commands::TransactionInfo * info) throw (   
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1397).

**6.202.3.36** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processSessionInfo  
(commands::SessionInfo * info) throw ( exceptions::ActiveMQException  
) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1397).

**6.202.3.37** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processShutdownInfo (com-  
mands::ShutdownInfo * info) throw ( exceptions::ActiveMQException )  
[pure virtual]`

**6.202.3.38** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processTransactionInfo  
(commands::TransactionInfo * info) throw (   
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1212).

**6.202.3.39**    virtual decaf::lang::Pointer<commands::Command>  
                activemq::state::CommandVisitor::processWireFormat  
                (commands::WireFormatInfo \* *info*) throw (  
                exceptions::ActiveMQException ) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/state/CommandVisitor.h

## 6.203 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p.1200) that returns NULL for all calls.

#include <src/main/activemq/state/CommandVisitorAdapter.h> Inheritance diagram for activemq::state::CommandVisitorAdapter:

### Public Member Functions

- virtual **~CommandVisitorAdapter** ()
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (commands::ConnectionId \*id AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (commands::SessionId \*id AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (commands::ProducerId \*id AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (commands::ConsumerId \*id AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (commands::DestinationInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (commands::DestinationInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (commands::RemoveSubscriptionInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (commands::Message \*send AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (commands::MessageAck \*ack AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (commands::MessagePull \*pull AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (commands::TransactionInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (commands::TransactionInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (commands::TransactionInfo \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )



- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** \*notification AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** \*ack AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** \*dispatch AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** \*replay AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** \*response AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processSessionInfo** (**commands::SessionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** \*info) throw ( exceptions::ActiveMQException )

### 6.203.1 Detailed Description

Default Implementation of a **CommandVisitor** (p.1200) that returns NULL for all calls.

Since:

3.0

## 6.203.2 Constructor & Destructor Documentation

- 6.203.2.1 virtual  
 activemq::state::CommandVisitorAdapter::~~CommandVisitorAdapter ()  
 [inline, virtual]

## 6.203.3 Member Function Documentation

- 6.203.3.1 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processBeginTransaction  
 (commands::TransactionInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.2 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processBrokerError  
 (commands::BrokerError \*error *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.3 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processBrokerInfo  
 (commands::BrokerInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.4 virtual decaf::lang::Pointer<commands::Command> ac-  
 tivemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase  
 (commands::TransactionInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.5 virtual decaf::lang::Pointer<commands::Command> ac-  
 tivemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase  
 (commands::TransactionInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.6 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processConnectionControl  
 (commands::ConnectionControl \*control *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.7 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processConnectionError  
 (commands::ConnectionError \*error *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.8 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processConnectionInfo  
 (commands::ConnectionInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.9 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processConsumerControl  
 (commands::ConsumerControl \*control *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]
- 6.203.3.10 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processConsumerInfo  
 (commands::ConsumerInfo \*info *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]

---

Generated on Wed Jul 25 23:57:04 2012 for 1.7.9 by doxygen

- 6.203.3.11 virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processControlCommand  
 (commands::ControlCommand \*command *AMQCPP\_UNUSED*) throw (  
 exceptions::ActiveMQException ) [inline, virtual]

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.203.3.30 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRemoveProducer`  
`(commands::ProducerId *id AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.31 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRemoveSession`  
`(commands::SessionId *id AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.32 `virtual decaf::lang::Pointer<commands::Command> ac-`  
`tivemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo`  
`(commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED)`  
`throw ( exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.33 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processReplayCommand`  
`(commands::ReplayCommand *replay AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.34 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processResponse`  
`(commands::Response *response AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.35 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRollbackTransaction`  
`(commands::TransactionInfo *info AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.36 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processSessionInfo`  
`(commands::SessionInfo *info AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.37 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processShutdownInfo`  
`(commands::ShutdownInfo *info AMQCPP_UNUSED) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.38 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processTransactionInfo`  
`(commands::TransactionInfo * info) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1206).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`,  
`activemq::core::ActiveMQConstants::TRANSACTION_STATE_`

```

COMMITONEPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
COMMITTWOPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_-
STATE_END,            activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
FORGET,               activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE,
activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER,      and
activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK.

```

```

6.203.3.39  virtual decaf::lang::Pointer<commands::Command>
               activemq::state::CommandVisitorAdapter::processWireFormat
               (commands::WireFormatInfo *info AMQCPP_UNUSED) throw (
               exceptions::ActiveMQException ) [inline, virtual]

```

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitorAdapter.h`

## 6.204 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

### Public Member Functions

- virtual **~Comparable** ()
- virtual int **compareTo** (const T &value) const =0  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const T &value) const =0  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.204.1 Detailed Description

```
template<typename T> class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

### 6.204.2 Constructor & Destructor Documentation

**6.204.2.1**    `template<typename T> virtual decaf::lang::Comparable< T >::~~Comparable () [inline, virtual]`

### 6.204.3 Member Function Documentation

**6.204.3.1**    `template<typename T> virtual int decaf::lang::Comparable< T >::compareTo (const T & value) const [pure virtual]`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementer must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p. 1214) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

**Parameters:**

*value* - the Object to be compared.

**Returns:**

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in `decaf::lang::Boolean` (p. 851), `decaf::lang::Byte` (p. 962), `decaf::lang::Character` (p. 1102), `decaf::lang::Double` (p. 1791), `decaf::lang::Float` (p. 1907), `decaf::lang::Integer` (p. 2080), `decaf::lang::Long` (p. 2423), and `decaf::lang::Short` (p. 3442).

### 6.204.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & value) const` [pure virtual]

**Returns:**

true if this value is considered equal to the passed value.

Implemented in `decaf::lang::Boolean` (p. 852), `decaf::lang::Byte` (p. 963), `decaf::lang::Character` (p. 1103), `decaf::lang::Double` (p. 1793), `decaf::lang::Float` (p. 1908), `decaf::lang::Integer` (p. 2081), `decaf::lang::Long` (p. 2424), and `decaf::lang::Short` (p. 3443).

### 6.204.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & value) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 852), `decaf::lang::Byte` (p. 964), `decaf::lang::Character` (p. 1105), `decaf::lang::Double` (p. 1795), `decaf::lang::Float` (p. 1910), `decaf::lang::Integer` (p. 2084), `decaf::lang::Long` (p. 2427), and `decaf::lang::Short` (p. 3444).

**6.204.3.4** `template<typename T> virtual bool decaf::lang::Comparable< T  
>::operator==(const T & value) const` [pure virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 853), `decaf::lang::Byte` (p. 965), `decaf::lang::Character` (p. 1105), `decaf::lang::Double` (p. 1795), `decaf::lang::Float` (p. 1911), `decaf::lang::Integer` (p. 2084), `decaf::lang::Long` (p. 2427), and `decaf::lang::Short` (p. 3445).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Comparable.h`



## 6.205 decaf::util::Comparator< T > Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

### Public Member Functions

- virtual `~Comparator()`
- virtual `bool operator()(const T &left, const T &right) const =0`  
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1217) to be passed to an STL **Map** (p. 2459) for use as the sorting criteria.*
- virtual `int compare(const T &o1, const T &o2) const =0`  
*Compares its two arguments for order.*

### 6.205.1 Detailed Description

```
template<typename T> class decaf::util::Comparator< T >
```

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as `Collections.sort`) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1217) *c* on a set of elements *S* is said to be consistent with equals if and only if ( `compare( e1, e2) == 0` ) has the same boolean value as ( `e1 == e2` ) for every *e1* and *e2* in *S*.

### 6.205.2 Constructor & Destructor Documentation

**6.205.2.1** `template<typename T> virtual decaf::util::Comparator< T >::~Comparator()` [inline, virtual]

### 6.205.3 Member Function Documentation

**6.205.3.1** `template<typename T> virtual int decaf::util::Comparator< T >::compare(const T & o1, const T & o2) const` [pure virtual]

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn( compare(x, y)) == -sgn(compare(y, x) )` for all *x* and *y*. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementor must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all *z*.

It is generally the case, but not strictly required that `(compare(x, y)==0) == ( x == y )`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

**Parameters:**

- o1* - the first object to be compared
- o2* - the second object to be compared

**Returns:**

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in **decaf::util::comparators::Less< E >** (p. 2328).

**6.205.3.2** `template<typename T> virtual bool decaf::util::Comparator< T >::operator() (const T & left, const T & right) const` [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1217) to be passed to an STL **Map** (p. 2459) for use as the sorting criteria.

**Parameters:**

- left* - the Left hand side operand.
- right* - the Right hand side operand.

**Returns:**

true if the value of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 2329).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

## 6.206 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

### Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > & getComponents** ()
- const **StlList< URI > & getComponents** () const
- void **setComponents** (const **StlList< URI > &components**)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties & getParameters** () const
- void **setParameters** (const **Properties &parameters**)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw ( decaf::net::URISyntaxException )

### 6.206.1 Detailed Description

Represents a Composite URI.

**Since:**

3.0

## 6.206.2 Constructor & Destructor Documentation

6.206.2.1 `activemq::util::CompositeData::CompositeData ()`

6.206.2.2 `virtual activemq::util::CompositeData::~~CompositeData ()` [virtual]

## 6.206.3 Member Function Documentation

6.206.3.1 `const StlList<URI>& activemq::util::CompositeData::getComponents ()`  
const [inline]

6.206.3.2 `StlList<URI>& activemq::util::CompositeData::getComponents ()`  
[inline]

6.206.3.3 `std::string activemq::util::CompositeData::getFragment ()` const  
[inline]

6.206.3.4 `std::string activemq::util::CompositeData::getHost ()` const [inline]

6.206.3.5 `const Properties& activemq::util::CompositeData::getParameters ()`  
const [inline]

6.206.3.6 `std::string activemq::util::CompositeData::getPath ()` const [inline]

6.206.3.7 `std::string activemq::util::CompositeData::getScheme ()` const [inline]

6.206.3.8 `void activemq::util::CompositeData::setComponents (const StlList< URI`  
> & *components*) [inline]

6.206.3.9 `void activemq::util::CompositeData::setFragment (const std::string &`  
*fragment*) [inline]

6.206.3.10 `void activemq::util::CompositeData::setHost (const std::string & host)`  
[inline]

6.206.3.11 `void activemq::util::CompositeData::setParameters (const Properties &`  
*parameters*) [inline]

6.206.3.12 `void activemq::util::CompositeData::setPath (const std::string & path)`  
[inline]

6.206.3.13 `void activemq::util::CompositeData::setScheme (const std::string &`  
*scheme*) [inline]

6.206.3.14 `URI activemq::util::CompositeData::toURI ()` const throw (  
`decaf::net::URISyntaxException`)

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

## 6.207 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1223).

#include <src/main/activemq/threads/CompositeTask.h>Inheritance diagram for activemq::threads::CompositeTask:

### Public Member Functions

- virtual `~CompositeTask ()`
- virtual `bool isPending () const =0`

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3734) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.*

### 6.207.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1223).

Since:

3.0

### 6.207.2 Constructor & Destructor Documentation

**6.207.2.1** `virtual activemq::threads::CompositeTask::~~CompositeTask () [inline, virtual]`

### 6.207.3 Member Function Documentation

**6.207.3.1** `virtual bool activemq::threads::CompositeTask::isPending () const [pure virtual]`

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3734) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since:

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p.757), `activemq::transport::failover::CloseTransportsTask` (p.1151), and `activemq::transport::failover::FailoverTransport` (p.1879).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

## 6.208 activemq::threads::CompositeTaskRunner Class Reference

A **Task** (p. 3734) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

#include <src/main/activemq/threads/CompositeTaskRunner.h> Inheritance diagram for activemq::threads::CompositeTaskRunner:

### Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (CompositeTask \*task)  
*Adds a new **CompositeTask** (p. 1221) to the Set of Tasks that this class manages.*
- void **removeTask** (CompositeTask \*task)  
*Removes a **CompositeTask** (p. 1221) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()  
*Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.*

### Protected Member Functions

- virtual void **run** ()  
*Run method - called by the Thread class in the context of the thread.*
- virtual bool **iterate** ()  
*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.208.1 Detailed Description

A **Task** (p. 3734) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Since:

3.0

## 6.208.2 Constructor & Destructor Documentation

**6.208.2.1** `activemq::threads::CompositeTaskRunner::CompositeTaskRunner ()`

**6.208.2.2** `virtual  
activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner ()  
[virtual]`

## 6.208.3 Member Function Documentation

**6.208.3.1** `void activemq::threads::CompositeTaskRunner::addTask (CompositeTask  
* task)`

Adds a new **CompositeTask** (p. 1221) to the Set of Tasks that this class manages.

### Parameters:

*task* - Pointer to a **CompositeTask** (p. 1221) instance.

**6.208.3.2** `virtual bool activemq::threads::CompositeTaskRunner::iterate ()  
[protected, virtual]`

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

### Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 3734).

**6.208.3.3** `void activemq::threads::CompositeTaskRunner::removeTask  
(CompositeTask * task)`

Removes a **CompositeTask** (p. 1221) that was added previously.

### Parameters:

*task* - Pointer to a **CompositeTask** (p. 1221) instance.

**6.208.3.4** `virtual void activemq::threads::CompositeTaskRunner::run ()  
[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3325).

**6.208.3.5** `virtual void activemq::threads::CompositeTaskRunner::shutdown ()  
[virtual]`

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3736).



### 6.208.3.6 virtual void activemq::threads::CompositeTaskRunner::shutdown (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters:

*timeout* - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 3736).

### 6.208.3.7 virtual void activemq::threads::CompositeTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3737).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

## 6.209 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3883) is a **Transport** (p. 3883) implementation that is composed of several Transports.

#include <src/main/activemq/transport/CompositeTransport.h> Inheritance diagram for activemq::transport::CompositeTransport:

### Public Member Functions

- virtual **~CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0  
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3883) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)=0  
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3883) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3883) should result in that **Transport** (p. 3883) being disposed of.*

### 6.209.1 Detailed Description

A Composite **Transport** (p. 3883) is a **Transport** (p. 3883) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 3883) exists for each URI that is composed or there could be many active Transports working at once.

Since:

3.0

### 6.209.2 Constructor & Destructor Documentation

- 6.209.2.1 virtual **activemq::transport::CompositeTransport::~~CompositeTransport** () [inline, virtual]

### 6.209.3 Member Function Documentation

- 6.209.3.1 virtual void **activemq::transport::CompositeTransport::addURI** (const **List**< **URI** > & *uris*) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3883) is a composite of.

Parameters:

*uris* The new URI set to add to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1876).

### 6.209.3.2 virtual void activemq::transport::CompositeTransport::removeURI (const List< URI > & *uris*) [pure virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3883) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3883) should result in that **Transport** (p. 3883) being disposed of.

#### Parameters:

*uris* The new URI set to remove to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1881).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

## 6.210 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 2459) type that provides additional **atomic** (p. 173) `putIfAbsent`, `remove`, and `replace` methods alongside the already available **Map** (p. 2459) interface.

#include <src/main/decaf/util/concurrent/ConcurrentMap.h> Inheritance diagram for `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >`:

### Public Member Functions

- virtual `~ConcurrentMap()`
- virtual `bool putIfAbsent (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )`

*If the specified key is not already associated with a value, associate it with the given value.*

- virtual `bool remove (const K &key, const V &value)=0`

*Remove entry for key only if currently mapped to given value.*

- virtual `bool replace (const K &key, const V &oldValue, const V &newValue)=0`

*Replace entry for key only if currently mapped to given value.*

- virtual `V replace (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )`

*Replace entry for key only if currently mapped to some value.*

### 6.210.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR> class
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >
```

Interface for a **Map** (p. 2459) type that provides additional **atomic** (p. 173) `putIfAbsent`, `remove`, and `replace` methods alongside the already available **Map** (p. 2459) interface.

Since:

1.0

## 6.210.2 Constructor & Destructor Documentation

**6.210.2.1** `template<typename K, typename V, typename COMPARATOR>  
virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR  
>::~ConcurrentMap () [inline, virtual]`

## 6.210.3 Member Function Documentation

**6.210.3.1** `template<typename K, typename V, typename COMPARATOR>  
virtual bool decaf::util::concurrent::ConcurrentMap< K, V,  
COMPARATOR >::putIfAbsent (const K & key, const V & value)  
throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]`

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

### Parameters:

***key*** The key to map the value to.

***value*** The value to map to the given key.

### Returns:

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

### Exceptions:

***UnsupportedOperationException*** if the put operation is not supported by this map

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1243), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1243).

**6.210.3.2** `template<typename K, typename V, typename COMPARATOR> virtual  
bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR  
>::remove (const K & key, const V & value) [pure virtual]`

Remove entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

**Parameters:**

*key* key with which the specified value is associated.

*value* value associated with the specified key.

**Returns:**

true if the value was removed, false otherwise

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1243), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1243), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1243).

**6.210.3.3** `template<typename K, typename V, typename COMPARATOR>  
virtual V decaf::util::concurrent::ConcurrentMap< K, V,  
COMPARATOR >::replace (const K & key, const V & value) throw (  
decaf::lang::exceptions::NoSuchElementException ) [pure virtual]`

Replace entry for key only if currently mapped to some value. Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

#### Parameters:

*key* key with which the specified value is associated.

*value* value to be associated with the specified key.

#### Returns:

copy of the previous value associated with specified key, or throws an NoSuchElementException if there was no mapping for key.

#### Exceptions:

*NoSuchElementException* if there was no previous mapping.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p.1244), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p.1244), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p.1244), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p.1244), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p.1244), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p.1244), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p.1244).

**6.210.3.4** template<typename K, typename V, typename COMPARATOR> virtual  
bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR  
>::replace (const K & *key*, const V & *oldValue*, const V & *newValue*)  
[pure virtual]

Replace entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

#### Parameters:

*key* key with which the specified value is associated.

*oldValue* value expected to be associated with the specified key.

*newValue* value to be associated with the specified key.

#### Returns:

true if the value was replaced

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1245), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1245), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1245), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1245), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1245), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1245), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1245).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`



## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h> Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

### Public Member Functions

- **ConcurrentStlMap** ()  
*Default constructor - does nothing.*
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const  
*Comparison, equality is dependent on the method of determining if the element are equal.*  
**Parameters:**  
*source* - **Map** (p. 2459) to compare to this one.  
**Returns:**  
*true if the **Map** (p. 2459) passed is equal in value to this one.*
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
- virtual void **copy** (const **ConcurrentStlMap** &source)  
*Copies the content of the source map into this map.*  
*Erases all existing data in this map.*  
**Parameters:**  
*source* The source object to copy from.
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*  
**Exceptions:**  
*UnsupportedOperationException* if this map is unmodifiable.
- virtual bool **containsKey** (const K &key) const  
*Indicates whether or this map contains a value for the given key.*  
**Parameters:**  
*key* The key to look up.  
**Returns:**  
*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const

*Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

**Parameters:**

*value* The Value to look up.

**Returns:**

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

**Returns:**

*if the **Map** (p. 2459) contains any element or not, TRUE or FALSE*

- virtual std::size\_t **size** () const

**Returns:**

*The number of elements (key/value pairs) in this map.*

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2459).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.*

**Parameters:**

*key* The search key.

**Returns:**

*A reference to the value for the given key.*

**Exceptions:**

**NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2459).

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2459).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.*

**Parameters:**

*key* The search key.

**Returns:**

*A {const} reference to the value for the given key.*

**Exceptions:**

**NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2459).

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

**Parameters:**

*key* The target key.

*value* The value to be set.

**Exceptions:**

**UnsupportedOperationException** if this map is unmodifiable.

- virtual void **putAll** (const **ConcurrentStlMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.*

**Parameters:**

*other A **Map** (p. 2459) instance whose elements are to all be inserted in this **Map** (p. 2459).*

**Exceptions:**

**UnsupportedOperationException** *If the implementing class does not support the putAll operation.*

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*

**Parameters:**

*key The search key.*

**Returns:**

*a copy of the element that was previously mapped to the given key*

**Exceptions:**

**NoSuchElementException** *if this key is not in the **Map** (p. 2459).*  
**UnsupportedOperationException** *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3439) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2155), **Set.remove** (p. 187), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.*

**Returns:**

*the entire set of keys in this map as a std::vector.*

- virtual std::vector< V > **values** () const

**Returns:**

*the entire set of values in this map as a std::vector.*

- bool **putIfAbsent** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*If the specified key is not already associated with a value, associate it with the given value.*

- bool **remove** (const K &key, const V &value)

*Remove entry for key only if currently mapped to given value.*

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

*Replace entry for key only if currently mapped to given value.*

- V **replace** (const K &key, const V &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Replace entry for key only if currently mapped to some value.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to **Lock** (p. 2375) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.211.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >
```

**Map** (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map. This version of **Map** (p. 2459) extends the **ConcurrentMap** (p. 1228) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since:

1.0

## 6.211.2 Constructor & Destructor Documentation

**6.211.2.1** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,  
COMPARATOR >::ConcurrentStlMap () [inline]`

Default constructor - does nothing.

**6.211.2.2** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,  
COMPARATOR >::ConcurrentStlMap (const ConcurrentStlMap< K, V,  
COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

**Parameters:**

*source* The source map.

**6.211.2.3** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K,  
V, COMPARATOR >::ConcurrentStlMap (const Map< K, V,  
COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

**Parameters:**

*source* The source map.

**6.211.2.4** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V,  
COMPARATOR >::~~ConcurrentStlMap () [inline, virtual]`

## 6.211.3 Member Function Documentation

**6.211.3.1** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<  
K, V, COMPARATOR >::clear () throw ( de-  
cafe::lang::exceptions::UnsupportedOperationException ) [inline,  
virtual]`

Removes all keys and values from this map.

**Exceptions:**

*UnsupportedOperationException* if this map is unmodifiable.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2460).

Referenced by **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** >::copy().

**6.211.3.2** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

#### Parameters:

*key* The key to look up.

#### Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2461).

Referenced by **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** >::equals(), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** >::putIfAbsent(), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** >::remove(), and **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** >::replace().

**6.211.3.3** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters:

*value* The Value to look up.

#### Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2462).

**6.211.3.4** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source) [inline, virtual]`

**6.211.3.5** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const ConcurrentStlMap< K, V, COMPARATOR > & source) [inline, virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

**Parameters:**

*source* The source object to copy from.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2463).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

**6.211.3.6** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

**6.211.3.7** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const ConcurrentStlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

**Parameters:**

*source* - `Map` (p. 2459) to compare to this one.

**Returns:**

true if the `Map` (p. 2459) passed is equal in value to this one.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2463).

**6.211.3.8** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & key) const throw ( lang::exceptions::NoSuchElementException ) [inline, virtual]`

Gets the value mapped to the specified key in the `Map` (p. 2459).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

**Parameters:**

*key* The search key.

**Returns:**

A {const} reference to the value for the given key.

**Exceptions:**

*NoSuchElementException* if the key requests doesn't exist in the **Map** (p. 2459).

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2463).

```
6.211.3.9  template<typename K, typename V, typename COMPARATOR =
           std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap<
           K, V, COMPARATOR >::get (const K & key) throw (
           lang::exceptions::NoSuchElementException ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2459).

If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

**Parameters:**

*key* The search key.

**Returns:**

A reference to the value for the given key.

**Exceptions:**

*NoSuchElementException* if the key requests doesn't exist in the **Map** (p. 2459).

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2464).

```
6.211.3.10 template<typename K, typename V, typename COMPARATOR =
           std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
           K, V, COMPARATOR >::isEmpty () const [inline, virtual]
```

**Returns:**

if the **Map** (p. 2459) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2465).

```
6.211.3.11 template<typename K, typename V, typename
           COMPARATOR = std::less<K>> virtual std::vector<K>
           decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
           >::keySet () const [inline, virtual]
```

Returns a **Set** (p. 3439) view of the mappings contained in this map.



The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2155), **Set.remove** (p. 187), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

**Returns:**

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2466).

**6.211.3.12** `template<typename K, typename V, typename  
 COMPARATOR = std::less<K>> virtual void  
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
 >::lock () throw ( decaf::lang::exceptions::RuntimeException ) [inline,  
 virtual]`

Locks the object.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3701).

**6.211.3.13** `template<typename K, typename V, typename  
 COMPARATOR = std::less<K>> virtual void  
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
 >::notify () throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
 virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3702).

**6.211.3.14** `template<typename K, typename V, typename  
 COMPARATOR = std::less<K>> virtual void  
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
 >::notifyAll () throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
 virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

***RuntimeException*** if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.211.3.15** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::put (const K & key, const V & value) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Sets the value for the specified key.

**Parameters:**

***key*** The target key.

***value*** The value to be set.

**Exceptions:**

***UnsupportedOperationException*** if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2467).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

**6.211.3.16** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

**6.211.3.17** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const ConcurrentStlMap< K, V, COMPARATOR > & other) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.

**Parameters:**

***other*** A **Map** (p. 2459) instance whose elements are to all be inserted in this **Map** (p. 2459).

**Exceptions:**

*UnsupportedOperationException* If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2468).

Referenced by **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()**.

**6.211.3.18** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V,  
COMPARATOR >::putIfAbsent (const K & key, const V & value)  
throw ( decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {  
    map.put( key, value );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

**Parameters:**

*key* The key to map the value to.

*value* The value to map to the given key.

**Returns:**

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

**Exceptions:**

*UnsupportedOperationException* if the put operation is not supported by this map

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1229).

**6.211.3.19** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V,  
COMPARATOR >::remove (const K & key, const V & value) [inline,  
virtual]`

Remove entry for key only if currently mapped to given value. Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
```

```

    map.remove( key );
    return true;
} else {
    return false;
}

```

except that the action is performed atomically.

#### Parameters:

*key* key with which the specified value is associated.

*value* value associated with the specified key.

#### Returns:

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1230).

**6.211.3.20** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

#### Parameters:

*key* The search key.

#### Returns:

a copy of the element that was previously mapped to the given key

#### Exceptions:

*NoSuchElementException* if this key is not in the **Map** (p. 2459).

*UnsupportedOperationException* if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2468).

**6.211.3.21** `template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & value) throw ( decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Replace entry for key only if currently mapped to some value. Acts as

```

if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};

```

except that the action is performed atomically.

#### Parameters:

*key* key with which the specified value is associated.

*value* value to be associated with the specified key.

#### Returns:

copy of the previous value associated with specified key, or throws an NoSuchElementException if there was no mapping for key.

#### Exceptions:

*NoSuchElementException* if there was no previous mapping.

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1230).

**6.211.3.22** `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue) [inline, virtual]`

Replace entry for key only if currently mapped to given value. Acts as

```

if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}

```

except that the action is performed atomically.

#### Parameters:

*key* key with which the specified value is associated.

*oldValue* value expected to be associated with the specified key.

*newValue* value to be associated with the specified key.

#### Returns:

true if the value was replaced

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1231).

**6.211.3.23** `template<typename K, typename V, typename  
COMPARATOR = std::less<K>> virtual std::size_t  
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
>::size () const [inline, virtual]`

**Returns:**

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2469).

**6.211.3.24** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<  
K, V, COMPARATOR >::tryLock () throw (  
decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to **Lock** (p. 2375) the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3704).

**6.211.3.25** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<  
K, V, COMPARATOR >::unlock () throw (  
decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3705).

**6.211.3.26** `template<typename K, typename V, typename  
COMPARATOR = std::less<K>> virtual std::vector<V>  
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
>::values () const [inline, virtual]`

**Returns:**

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2470).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values()`.

**6.211.3.27** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE  
*nanos* additional time in nanoseconds with a range of 0-999999

**Exceptions:**

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]  
*RuntimeException* if an error occurs while waiting on the object.  
*InterruptedException* if the wait is interrupted before it completes.  
*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.211.3.28** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.  
*InterruptedException* if the wait is interrupted before it completes.  
*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

```
6.211.3.29  template<typename K, typename V, typename
            COMPARATOR = std::less<K>> virtual void
            decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::wait () throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

#### Exceptions:

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**



## 6.212 decaf::util::concurrent::locks::Condition Class Reference

**Condition** (p. 1249) factors out the **Mutex** (p. 2782) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2377) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

### Public Member Functions

- virtual `~Condition` ()
- virtual void **await** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted.*
- virtual void **awaitUninterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signalled.*
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )
- virtual void **signal** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up one waiting thread.*
- virtual void **signalAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up all waiting threads.*

### 6.212.1 Detailed Description

**Condition** (p. 1249) factors out the **Mutex** (p. 2782) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2377) implementations. Where a **Lock** (p. 2377) replaces the use of synchronized statements, a **Condition** (p. 1249) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1249) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1249) instance for a particular **Lock** (p. 2377) instance use its `newCondition()` method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1249) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1251); items[putptr] = x; if ( ++putptr == 100 ) putptr = 0; ++count; notEmpty->signal() (p. 1254); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1251); Object x = items[takeptr]; if ( ++takeptr == 100 ) takeptr = 0; --count; notFull->signal() (p. 1254); return x; } catch(...) { lock->unlock(); } }
```

(The `ArrayBlockingQueue` class provides this functionality, so there is no reason to implement this sample usage class.)

#### Implementation Considerations

When waiting upon a **Condition** (p. 1249), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 1249) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

## 6.212.2 Constructor & Destructor Documentation

**6.212.2.1** virtual decaf::util::concurrent::locks::Condition::~~Condition () [inline, virtual]

## 6.212.3 Member Function Documentation

**6.212.3.1** virtual bool decaf::util::concurrent::locks::Condition::await (long long *time*, const TimeUnit & *unit*) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses. This method is behaviorally equivalent to:

```
awaitNanos(unit.toNanos(time)) > 0
```

**Parameters:**

*time* - the maximum time to wait

*unit* - the time unit of the time argument

**Returns:**

false if the waiting time detectably elapsed before return from the method, else true

**Exceptions:**

**RuntimeException** if an unexpected error occurs while trying to wait on the **Condition** (p. 1249).

**InterruptedException** if the current thread is interrupted (and interruption of thread suspension is supported)

**IllegalMonitorStateException** if the caller is not the lock owner.

**6.212.3.2** virtual void decaf::util::concurrent::locks::Condition::await () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]

Causes the current thread to wait until it is signaled or interrupted. The lock associated with this **Condition** (p. 1249) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- \* Some other thread invokes the **signal()** (p. 1254) method for this **Condition** (p. 1249) and the current thread happens to be chosen as the thread to be awakened; or
- \* Some other thread invokes the **signalAll()** (p. 1254) method for this **Condition** (p. 1249); or
- \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

#### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1249) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

#### Exceptions:

***RuntimeException*** if an unexpected error occurs while trying to wait on the **Condition** (p. 1249).

***InterruptedException*** if the current thread is interrupted (and interruption of thread suspension is supported)

***IllegalMonitorStateException*** if the caller is not the lock owner.

```
6.212.3.3  virtual long long decaf::util::concurrent::locks::Condition::awaitNanos
            (long long nanosTimeout) throw ( de-
            caf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::InterruptedException, de-
            caf::lang::exceptions::IllegalMonitorStateException ) [pure
            virtual]
```

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

\* Some other thread invokes the **signal()** (p. 1254) method for this **Condition** (p. 1249) and the current thread happens to be chosen as the thread to be awakened; or \* Some other thread invokes the **signalAll()** (p. 1254) method for this **Condition** (p. 1249); or \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or \* The specified waiting time elapses; or \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied `nanosTimeout` value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanosTimeout =
unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0) nanosTimeout =
theCondition->awaitNanos(nanosTimeout); else return false; } // ... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

#### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p.1249) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

#### Parameters:

*nanosTimeout* - the maximum time to wait, in nanoseconds

#### Returns:

an estimate of the `nanosTimeout` value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

#### Exceptions:

**RuntimeException** if an unexpected error occurs while trying to wait on the **Condition** (p.1249).

**InterruptedException** if the current thread is interrupted (and interruption of thread suspension is supported)

**IllegalMonitorStateException** if the caller is not the lock owner.

**6.212.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly()** **throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )** [pure virtual]

Causes the current thread to wait until it is signalled. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes the **signal()** (p.1254) method for this **Condition** (p.1249) and the current thread happens to be chosen as the thread to be awakened; or \* Some other thread invokes the **signalAll()** (p.1254) method for this **Condition** (p.1249); or \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

#### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1249) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

#### Exceptions:

***RuntimeException*** if an unexpected error occurs while trying to wait on the **Condition** (p. 1249).

***IllegalMonitorStateException*** if the caller is not the lock owner.

**6.212.3.5** `virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & deadline) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )` [pure virtual]

**6.212.3.6** `virtual void decaf::util::concurrent::locks::Condition::signal () throw ( decaf::lang::exceptions::RuntimeException )` [pure virtual]

Wakes up one waiting thread. If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

#### Exceptions:

***RuntimeException*** if an unexpected error occurs while trying to wait on the **Condition** (p. 1249).

**6.212.3.7** `virtual void decaf::util::concurrent::locks::Condition::signalAll () throw ( decaf::lang::exceptions::RuntimeException )` [pure virtual]

Wakes up all waiting threads. If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

#### Exceptions:

***RuntimeException*** if an unexpected error occurs while trying to wait on the **Condition** (p. 1249).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Condition.h`

## 6.213 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

### Public Member Functions

- **ConditionHandle** ()
- **~ConditionHandle** ()
- **ConditionHandle** ()
- **~ConditionHandle** ()

### Data Fields

- pthread\_cond\_t **condition**
- MutexHandle \* **mutex**
- HANDLE semaphore
- CRITICAL\_SECTION **criticalSection**
- volatile unsigned int **numWaiting**
- volatile unsigned int **numWake**
- volatile unsigned int **generation**

### 6.213.1 Constructor & Destructor Documentation

6.213.1.1 decaf::util::concurrent::ConditionHandle::ConditionHandle () [inline]

6.213.1.2 decaf::util::concurrent::ConditionHandle::~~ConditionHandle () [inline]

6.213.1.3 decaf::util::concurrent::ConditionHandle::ConditionHandle () [inline]

6.213.1.4 decaf::util::concurrent::ConditionHandle::~~ConditionHandle () [inline]

### 6.213.2 Field Documentation

6.213.2.1 pthread\_cond\_t decaf::util::concurrent::ConditionHandle::condition

6.213.2.2 CRITICAL\_SECTION decaf::util::concurrent::ConditionHandle::criticalSection

6.213.2.3 volatile unsigned int decaf::util::concurrent::ConditionHandle::generation

6.213.2.4 MutexHandle \* decaf::util::concurrent::ConditionHandle::mutex

6.213.2.5 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting

6.213.2.6 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake

6.213.2.7 HANDLE decaf::util::concurrent::ConditionHandle::semaphore

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h`
- `src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h`



## 6.214 decaf::internal::util::concurrent::ConditionImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```

### Static Public Member Functions

- static **decaf::util::concurrent::ConditionHandle** \* **create** (decaf::util::concurrent::MutexHandle \*mutex)  
*Creates the Condition object and attaches it to the given MutexHandle.*
- static void **destroy** (decaf::util::concurrent::ConditionHandle \*handle)  
*Destroy a previously create Condition instance.*
- static void **wait** (decaf::util::concurrent::ConditionHandle \*condition)  
*Waits for the condition to be signaled.*
- static void **wait** (decaf::util::concurrent::ConditionHandle \*condition, long long mills, long long nanos)  
*Waits for the condition to be signaled or for the time specified to ellapse.*
- static void **notify** (decaf::util::concurrent::ConditionHandle \*condition)  
*Signals one Thread that is waiting on this condition to wake up.*
- static void **notifyAll** (decaf::util::concurrent::ConditionHandle \*condition)  
*Signals all Threads that is waiting on this condition to wake up.*

### 6.214.1 Member Function Documentation

**6.214.1.1** static **decaf::util::concurrent::ConditionHandle\***  
**decaf::internal::util::concurrent::ConditionImpl::create**  
(decaf::util::concurrent::MutexHandle \* *mutex*) [static]

Creates the Condition object and attaches it to the given MutexHandle.

#### Parameters:

*mutex* the Mutex handle that this Condition is attached to.

#### Returns:

a newly constructed Condition handle that is attached to the given handle.

**6.214.1.2** static void **decaf::internal::util::concurrent::ConditionImpl::destroy**  
(decaf::util::concurrent::ConditionHandle \* *handle*) [static]

Destroy a previously create Condition instance.

#### Parameters:

*handle* The Condition handle to be destroyed.

**6.214.1.3** `static void decaf::internal::util::concurrent::ConditionImpl::notify`  
(`decaf::util::concurrent::ConditionHandle * condition`) [static]

Signals one Thread that is waiting on this condition to wake up.

**Parameters:**

*condition* the handle to the condition to wait on.

**6.214.1.4** `static void decaf::internal::util::concurrent::ConditionImpl::notifyAll`  
(`decaf::util::concurrent::ConditionHandle * condition`) [static]

Signals all Threads that is waiting on this condition to wake up.

**Parameters:**

*condition* the handle to the condition to wait on.

**6.214.1.5** `static void decaf::internal::util::concurrent::ConditionImpl::wait`  
(`decaf::util::concurrent::ConditionHandle * condition, long long mills,  
long long nanos`) [static]

Waits for the condition to be signaled or for the time specified to elapse.

**Parameters:**

*condition* the handle to the condition to wait on.

*mills* the time in milliseconds to wait for the condition to be signaled.

*nanos* additional time in nanoseconds to wait for the thread to be signaled.

**6.214.1.6** `static void decaf::internal::util::concurrent::ConditionImpl::wait`  
(`decaf::util::concurrent::ConditionHandle * condition`) [static]

Waits for the condition to be signaled.

**Parameters:**

*condition* the handle to the condition to wait on.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/ConditionImpl.h`

## 6.215 decaf::net::ConnectException Class Reference

#include <src/main/decaf/net/ConnectException.h> Inheritance diagram for decaf::net::ConnectException:

### Public Member Functions

- **ConnectException** () throw ()  
*Default Constructor.*
- **ConnectException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ConnectException** (const **ConnectException** &ex) throw ()  
*Copy Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ConnectException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ConnectException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ConnectException** () throw ()

### 6.215.1 Constructor & Destructor Documentation

#### 6.215.1.1 decaf::net::ConnectException::ConnectException () throw () [inline]

Default Constructor.

#### 6.215.1.2 decaf::net::ConnectException::ConnectException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.215.1.3 `decaf::net::ConnectException::ConnectException (const ConnectException & ex) throw ()` [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.215.1.4 `decaf::net::ConnectException::ConnectException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.215.1.5 `decaf::net::ConnectException::ConnectException (const std::exception * cause) throw ()` [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.215.1.6 `decaf::net::ConnectException::ConnectException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.215.1.7** virtual decaf::net::ConnectException::~~ConnectException () throw ()  
[inline, virtual]

## 6.215.2 Member Function Documentation

**6.215.2.1** virtual ConnectException\* decaf::net::ConnectException::clone () const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3522).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ConnectException.h**

## 6.216 cms::Connection Class Reference

The client's connection to its provider.

#include <src/main/cms/Connection.h> Inheritance diagram for cms::Connection:

### Public Member Functions

- virtual **~Connection** ()
- virtual void **close** ()=0 throw ( CMSEException )  
*Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).*
- virtual const **ConnectionMetaData** \* **getMetaData** () const =0 throw ( CMSEException )  
*Gets the metadata for this connection.*
- virtual **Session** \* **createSession** ()=0 throw ( CMSEException )  
*Creates an **AUTO\_ACKNOWLEDGE Session** (p. 3365).*
- virtual **Session** \* **createSession** (**Session::AcknowledgeMode** ackMode)=0 throw ( CMSEException )  
*Creates a new **Session** (p. 3365) to work for this **Connection** (p. 1262) using the specified acknowledgment mode.*
- virtual std::string **getClientID** () const =0  
*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the **setClientID** method.*
- virtual void **setClientID** (const std::string &clientID)=0  
*Sets the client identifier for this connection.*
- virtual **ExceptionListener** \* **getExceptionListener** () const =0  
*Gets the registered Exception Listener for this connection.*
- virtual void **setExceptionListener** (**ExceptionListener** \*listener)=0  
*Sets the registered Exception Listener for this connection.*

### 6.216.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.

- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1385) object.
- It supports an optional **ExceptionListener** (p. 1838) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

#### Since:

1.0

## 6.216.2 Constructor & Destructor Documentation

### 6.216.2.1 virtual cms::Connection::~~Connection () [inline, virtual]

## 6.216.3 Member Function Documentation

### 6.216.3.1 virtual void cms::Connection::close () throw ( CMSException ) [pure virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

#### Exceptions:

*CMSException* (p. 1160)

Implements **cms::Closeable** (p. 1148).

Implemented in **activemq::core::ActiveMQConnection** (p. 279).

**6.216.3.2** `virtual Session* cms::Connection::createSession  
(Session::AcknowledgeMode ackMode) throw ( CMSException ) [pure  
virtual]`

Creates a new **Session** (p. 3365) to work for this **Connection** (p. 1262) using the specified acknowledgment mode.

**Parameters:**

*ackMode* the Acknowledgment Mode to use.

**Exceptions:**

*CMSException* (p. 1160)

Implemented in **activemq::core::ActiveMQConnection** (p. 279).

**6.216.3.3** `virtual Session* cms::Connection::createSession () throw ( CMSException  
) [pure virtual]`

Creates an AUTO\_ACKNOWLEDGE **Session** (p. 3365).

**Exceptions:**

*CMSException* (p. 1160)

Implemented in **activemq::core::ActiveMQConnection** (p. 279).

**6.216.3.4** `virtual std::string cms::Connection::getClientID () const [pure virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

**Returns:**

Client Id String for this **Connection** (p. 1262).

**Exceptions:**

*CMSException* (p. 1160) if the provider fails to return the client id or an internal error occurs.

Implemented in **activemq::core::ActiveMQConnection** (p. 281).

**6.216.3.5** `virtual ExceptionListener* cms::Connection::getExceptionListener ()  
const [pure virtual]`

Gets the registered Exception Listener for this connection.

**Returns:**

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 281).



### 6.216.3.6 virtual const ConnectionMetaData\* cms::Connection::getMetaData () const throw ( CMSEException ) [pure virtual]

Gets the metadata for this connection.

**Returns:**

the connection MetaData pointer ( caller does not own it ).

**Exceptions:**

*CMSEException* (p. 1160) if the provider fails to get the connection metadata for this connection.

**See also:**

**ConnectionMetaData** (p. 1385)

**Since:**

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 282).

### 6.216.3.7 virtual void cms::Connection::setClientID (const std::string & clientID) [pure virtual]

Sets the client identifier for this connection. The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1324) object and transparently assigned to the **Connection** (p. 1262) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1997).

**Parameters:**

*clientID* The unique client identifier to assign to the **Connection** (p. 1262).

**Exceptions:**

*CMSEException* (p. 1160) if the provider fails to set the client id due to some internal error.

*InvalidClientIDException* if the id given is somehow invalid or is a duplicate.

*IllegalStateException* (p. 1997) if the client tries to set the id after a **Connection** (p. 1262) method has been called.

Implemented in **activemq::core::ActiveMQConnection** (p. 287).

### 6.216.3.8 virtual void cms::Connection::setExceptionListener (ExceptionListener \* listener) [pure virtual]

Sets the registered Exception Listener for this connection.

**Parameters:**

*listener* pointer to and ExceptionListener (p. 1838)

Implemented in **activemq::core::ActiveMQConnection** (p. 288).

The documentation for this class was generated from the following file:

- `src/main/cms/Connection.h`

## 6.217 activemq::commands::ConnectionControl Class Reference

#include <src/main/activemq/commands/ConnectionControl.h> Inheritance diagram for activemq::commands::ConnectionControl:

### Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionControl** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isResume** () const
- virtual void **setResume** (bool resume)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool suspend)
- virtual const std::string & **getConnectedBrokers** () const
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (const std::string &connectedBrokers)
- virtual const std::string & **getReconnectTo** () const
- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (const std::string &reconnectTo)
- virtual bool **isRebalanceConnection** () const
- virtual void **setRebalanceConnection** (bool rebalanceConnection)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_CONNECTIONCONTROL** = 18

## Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

### 6.217.1 Constructor & Destructor Documentation

**6.217.1.1** `activemq::commands::ConnectionControl::ConnectionControl ()`

**6.217.1.2** `virtual activemq::commands::ConnectionControl::~~ConnectionControl ()`  
[virtual]

### 6.217.2 Member Function Documentation

**6.217.2.1** `virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure () const`  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.217.2.2** `virtual void activemq::commands::ConnectionControl::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

### 6.217.2.3 virtual bool activemq::commands::ConnectionControl::equals (const DataStructure \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

### 6.217.2.4 virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers () [virtual]

### 6.217.2.5 virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers () const [virtual]

### 6.217.2.6 virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.217.2.7 `virtual std::string& activemq::commands::ConnectionControl::getReconnectTo ()`  
[virtual]
- 6.217.2.8 `virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo ()`  
`const` [virtual]
- 6.217.2.9 `virtual bool activemq::commands::ConnectionControl::isClose () const`  
[virtual]
- 6.217.2.10 `virtual bool activemq::commands::ConnectionControl::isExit () const`  
[virtual]
- 6.217.2.11 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant () const` [virtual]
- 6.217.2.12 `virtual bool activemq::commands::ConnectionControl::isRebalanceConnection () const`  
[virtual]
- 6.217.2.13 `virtual bool activemq::commands::ConnectionControl::isResume ()`  
`const` [virtual]
- 6.217.2.14 `virtual bool activemq::commands::ConnectionControl::isSuspend ()`  
`const` [virtual]
- 6.217.2.15 `virtual void activemq::commands::ConnectionControl::setClose (bool`  
`close)` [virtual]
- 6.217.2.16 `virtual void activemq::commands::ConnectionControl::setConnectedBrokers (const`  
`std::string & connectedBrokers)` [virtual]
- 6.217.2.17 `virtual void activemq::commands::ConnectionControl::setExit (bool`  
`exit)` [virtual]
- 6.217.2.18 `virtual void activemq::commands::ConnectionControl::setFaultTolerant`  
`(bool faultTolerant)` [virtual]
- 6.217.2.19 `virtual void activemq::commands::ConnectionControl::setRebalanceConnection (bool`  
`rebalanceConnection)` [virtual]
- 6.217.2.20 `virtual void activemq::commands::ConnectionControl::setReconnectTo`  
`(const std::string & reconnectTo)` [virtual]
- 6.217.2.21 `virtual void activemq::commands::ConnectionControl::setResume (bool`  
`resume)` [virtual]
- 6.217.2.22 `virtual void activemq::commands::ConnectionControl::setSuspend (bool`  
`suspend)` [virtual]
- 6.217.2.23 `virtual std::string activemq::commands::ConnectionControl::toString ()`  
`const` [virtual]

---

Generated on Wed Jul 25 23:57:04 2012 for activemq-cpp-3.2.5 by Doxygen

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

```
6.217.2.24 virtual Pointer<Command> ac-
    tivemq::commands::ConnectionControl::visit
    (activemq::state::CommandVisitor * visitor) throw (
    exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1198).

**6.217.3 Field Documentation**

- 6.217.3.1** `bool activemq::commands::ConnectionControl::close` [protected]
- 6.217.3.2** `std::string activemq::commands::ConnectionControl::connectedBrokers` [protected]
- 6.217.3.3** `bool activemq::commands::ConnectionControl::exit` [protected]
- 6.217.3.4** `bool activemq::commands::ConnectionControl::faultTolerant` [protected]
- 6.217.3.5** `const unsigned char activemq::commands::ConnectionControl::ID_ - CONNECTIONCONTROL = 18` [static]
- 6.217.3.6** `bool activemq::commands::ConnectionControl::rebalanceConnection` [protected]
- 6.217.3.7** `std::string activemq::commands::ConnectionControl::reconnectTo` [protected]
- 6.217.3.8** `bool activemq::commands::ConnectionControl::resume` [protected]
- 6.217.3.9** `bool activemq::commands::ConnectionControl::suspend` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

## 6.218 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1272).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.218.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1272).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.218.2 Constructor & Destructor Documentation

**6.218.2.1** `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller()` `[inline]`

**6.218.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` `[inline, virtual]`

## 6.218.3 Member Function Documentation

**6.218.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.218.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.218.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.218.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.218.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.218.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.218.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h

## 6.219 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1276).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.219.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **ConnectionControlMarshaller** (p. 1276).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.219.2 Constructor & Destructor Documentation

**6.219.2.1** `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller()` `[inline]`

**6.219.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` `[inline, virtual]`

## 6.219.3 Member Function Documentation

**6.219.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.219.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.219.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.219.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.219.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.219.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.219.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h

## 6.220 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1280).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller`:

### Public Member Functions

- `ConnectionControlMarshaller ()`
- `virtual ~ConnectionControlMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.220.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1280).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.220.2 Constructor & Destructor Documentation

**6.220.2.1** `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

**6.220.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

## 6.220.3 Member Function Documentation

**6.220.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.220.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.220.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.220.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.220.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.220.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.220.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h

## 6.221 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1284).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller`:

### Public Member Functions

- `ConnectionControlMarshaller ()`
- `virtual ~ConnectionControlMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.221.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1284).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.221.2 Constructor & Destructor Documentation

**6.221.2.1** `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

**6.221.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

## 6.221.3 Member Function Documentation

**6.221.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.221.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.221.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.221.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.221.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.221.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.221.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h

## 6.222 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1288).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller`:

### Public Member Functions

- `ConnectionControlMarshaller ()`
- `virtual ~ConnectionControlMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.222.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1288).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.222.2 Constructor & Destructor Documentation

**6.222.2.1** `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::ConnectionControlMarshaller()` `[inline]`

**6.222.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` `[inline, virtual]`

## 6.222.3 Member Function Documentation

**6.222.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.222.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.222.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.222.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.222.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.222.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.222.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h

## 6.223 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1292).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller`:

### Public Member Functions

- `ConnectionControlMarshaller ()`
- `virtual ~ConnectionControlMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.223.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionControlMarshaller` (p. 1292).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.223.2 Constructor & Destructor Documentation

**6.223.2.1** `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

**6.223.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

## 6.223.3 Member Function Documentation

**6.223.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.223.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.223.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.223.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.223.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.223.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.223.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h

## 6.224 activemq::commands::ConnectionError Class Reference

#include <src/main/activemq/commands/ConnectionError.h> Inheritance diagram for activemq::commands::ConnectionError:

### Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionError** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONERROR** = 16

### Protected Attributes

- **Pointer**< **BrokerError** > exception
- **Pointer**< **ConnectionId** > connectionId



## 6.224.1 Constructor & Destructor Documentation

**6.224.1.1** `activemq::commands::ConnectionError::ConnectionError ()`

**6.224.1.2** `virtual activemq::commands::ConnectionError::~~ConnectionError ()`  
[virtual]

## 6.224.2 Member Function Documentation

**6.224.2.1** `virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.224.2.2** `virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.224.2.3** `virtual bool activemq::commands::ConnectionError::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.224.2.4 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ()`  
[virtual]
- 6.224.2.5 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ()`  
`const` [virtual]
- 6.224.2.6 `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.224.2.7 `virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException ()`  
[virtual]
- 6.224.2.8 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException ()`  
`const` [virtual]
- 6.224.2.9 `virtual void activemq::commands::ConnectionError::setConnectionId`  
`(const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.224.2.10 `virtual void activemq::commands::ConnectionError::setException (const`  
`Pointer< BrokerError > & exception)` [virtual]
- 6.224.2.11 `virtual std::string activemq::commands::ConnectionError::toString ()`  
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.224.2.12 `virtual Pointer<Command> activemq::commands::ConnectionError::visit`  
`(activemq::state::CommandVisitor * visitor) throw (`  
`exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

- a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

**6.224.3 Field Documentation**

**6.224.3.1** **Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId**  
[protected]

**6.224.3.2** **Pointer<BrokerError> activemq::commands::ConnectionError::exception**  
[protected]

**6.224.3.3** **const unsigned char activemq::commands::ConnectionError::ID\_ - CONNECTIONERROR = 16** [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionError.h`

## 6.225 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1300).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.225.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1300).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.225.2 Constructor & Destructor Documentation

**6.225.2.1** `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.225.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.225.3 Member Function Documentation

**6.225.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.225.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.225.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.225.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.225.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.225.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.225.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h

## 6.226 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionErrorMarshaller` (p. 1304).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller`:

### Public Member Functions

- `ConnectionErrorMarshaller ()`
- `virtual ~ConnectionErrorMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.226.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `ConnectionErrorMarshaller` (p. 1304).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.226.2 Constructor & Destructor Documentation

**6.226.2.1** `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.226.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.226.3 Member Function Documentation

**6.226.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.226.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.226.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.226.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.226.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.226.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.226.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

## 6.227 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1308).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.227.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1308).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.227.2 Constructor & Destructor Documentation

**6.227.2.1** `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.227.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.227.3 Member Function Documentation

**6.227.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.227.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.227.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.227.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.227.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.227.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.227.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h

## 6.228 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1312).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.228.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1312).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.228.2 Constructor & Destructor Documentation

**6.228.2.1** `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.228.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.228.3 Member Function Documentation

**6.228.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.228.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.228.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.228.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.228.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.228.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.228.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

## 6.229 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1316).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.229.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1316).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.229.2 Constructor & Destructor Documentation

**6.229.2.1** `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.229.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.229.3 Member Function Documentation

**6.229.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.229.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.229.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.229.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.229.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.229.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.229.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h

## 6.230 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1320).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.230.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1320).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.230.2 Constructor & Destructor Documentation

**6.230.2.1** `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

**6.230.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

## 6.230.3 Member Function Documentation

**6.230.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.230.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.230.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.230.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.230.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.230.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.230.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h

## 6.231 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p.1262) objects returned implement the CMS **Connection** (p.1262) interface and hide the CMS Provider specific implementation details behind that interface.

#include <src/main/cms/ConnectionFactory.h> Inheritance diagram for cms::ConnectionFactory:

### Public Member Functions

- virtual **~ConnectionFactory** ()
- virtual **Connection \* createConnection** ()=0 throw ( CMSEException )  
*Creates a connection with the default user identity.*
- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password)=0 throw ( cms::CMSEException )  
*Creates a connection with the default specified identity.*
- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw ( cms::CMSEException )  
*Creates a connection with the specified user identity.*

### Static Public Member Functions

- static **ConnectionFactory \* createCMSConnectionFactory** (const std::string &brokerURI) throw ( cms::CMSEException )  
*Static method that is used to create a provider specific connection factory.*

#### 6.231.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p.1262) objects returned implement the CMS **Connection** (p.1262) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p.1324) either directly by instantiating the provider specific implementation of the factory or by using the static method **createCMSConnectionFactory** which all providers are required to implement.

Since:

1.0

## 6.231.2 Constructor & Destructor Documentation

**6.231.2.1** `virtual cms::ConnectionFactory::~~ConnectionFactory () [inline, virtual]`

## 6.231.3 Member Function Documentation

**6.231.3.1** `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & brokerURI) throw ( cms::CMSException ) [static]`

Static method that is used to create a provider specific connection factory. The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p.1324) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

### Parameters:

***brokerURI*** The remote address to use to connect to the Provider.

### Returns:

A pointer to a provider specific implementation of the **ConnectionFactory** (p.1324) interface, the caller is responsible for deleting this resource.

### Exceptions:

***CMSException*** (p. 1160) if an internal error occurs while creating the **ConnectionFactory** (p.1324).

**6.231.3.2** `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw ( cms::CMSException ) [pure virtual]`

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3586) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

### Parameters:

***username*** The user name used to authenticate with the Provider.

***password*** The password used to authenticate with the Provider.

***clientId*** The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

### Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

### Exceptions:

***CMSException*** (p. 1160) if an internal error occurs while creating the **Connection** (p.1262).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 295).

**6.231.3.3** `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password) throw ( cms::CMSEException )` [pure virtual]

Creates a connection with the default specified identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3586) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

**Parameters:**

*username* The user name used to authenticate with the Provider.

*password* The password used to authenticate with the Provider.

**Returns:**

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

**Exceptions:**

*CMSEException* (p. 1160) if an internal error occurs while creating the **Connection** (p. 1262).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 296).

**6.231.3.4** `virtual Connection* cms::ConnectionFactory::createConnection () throw ( CMSEException )` [pure virtual]

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3586) method is explicitly called.

**Returns:**

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

**Exceptions:**

*CMSEException* (p. 1160) if an internal error occurs while creating the **Connection** (p. 1262).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 296).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

## 6.232 activemq::commands::ConnectionId Class Reference

#include <src/main/activemq/commands/ConnectionId.h> Inheritance diagram for activemq::commands::ConnectionId:

### Public Types

- typedef decaf::lang::PointerComparator< ConnectionId > COMPARATOR

### Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** \*sessionId)
- **ConnectionId** (const **ProducerId** \*producerId)
- **ConnectionId** (const **ConsumerId** \*consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **ConnectionId** &value) const
- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONID** = 120

## Protected Attributes

- `std::string` `value`

### 6.232.1 Member Typedef Documentation

**6.232.1.1** `typedef decaf::lang::PointerComparator<ConnectionId>`  
`activemq::commands::ConnectionId::COMPARATOR`

### 6.232.2 Constructor & Destructor Documentation

**6.232.2.1** `activemq::commands::ConnectionId::ConnectionId ()`

**6.232.2.2** `activemq::commands::ConnectionId::ConnectionId (const ConnectionId`  
`& other)`

**6.232.2.3** `activemq::commands::ConnectionId::ConnectionId (const SessionId *`  
`sessionId)`

**6.232.2.4** `activemq::commands::ConnectionId::ConnectionId (const ProducerId *`  
`producerId)`

**6.232.2.5** `activemq::commands::ConnectionId::ConnectionId (const ConsumerId *`  
`consumerId)`

**6.232.2.6** `virtual activemq::commands::ConnectionId::~~ConnectionId ()` [virtual]

### 6.232.3 Member Function Documentation

**6.232.3.1** `virtual ConnectionId* ac-`  
`tivemq::commands::ConnectionId::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.232.3.2** `virtual int activemq::commands::ConnectionId::compareTo (const`  
`ConnectionId & value) const` [virtual]

**6.232.3.3** `virtual void activemq::commands::ConnectionId::copyDataStructure`  
`(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object



**6.232.3.4** virtual bool activemq::commands::ConnectionId::equals (const ConnectionId & *value*) const [virtual]

**6.232.3.5** virtual bool activemq::commands::ConnectionId::equals (const DataStructure \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

**6.232.3.6** virtual unsigned char activemq::commands::ConnectionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

**6.232.3.7** virtual std::string& activemq::commands::ConnectionId::getValue () [virtual]

**6.232.3.8** virtual const std::string& activemq::commands::ConnectionId::getValue () const [virtual]

**6.232.3.9** virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & *value*) const [virtual]

**6.232.3.10** ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & *other*)

**6.232.3.11** virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & *value*) const [virtual]

**6.232.3.12** virtual void activemq::commands::ConnectionId::setValue (const std::string & *value*) [virtual]

**6.232.3.13** virtual std::string activemq::commands::ConnectionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 833).

## 6.232.4 Field Documentation

**6.232.4.1** `const unsigned char activemq::commands::ConnectionId::ID_ - CONNECTIONID = 120` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

**6.232.4.2** `std::string activemq::commands::ConnectionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

## 6.233 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1331).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.233.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1331).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.233.2 Constructor & Destructor Documentation

**6.233.2.1** `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.233.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.233.3 Member Function Documentation

**6.233.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.233.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.233.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.233.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.233.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.233.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.233.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

## 6.234 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1335).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.234.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1335).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.234.2 Constructor & Destructor Documentation

**6.234.2.1** `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.234.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.234.3 Member Function Documentation

**6.234.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.234.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.234.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.234.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.234.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.234.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.234.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h

## 6.235 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1339).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.235.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1339).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.235.2 Constructor & Destructor Documentation

**6.235.2.1** `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.235.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.235.3 Member Function Documentation

**6.235.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.235.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.235.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.235.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.235.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.235.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.235.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

## 6.236 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1343).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.236.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1343).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.236.2 Constructor & Destructor Documentation

**6.236.2.1** `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.236.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.236.3 Member Function Documentation

**6.236.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.236.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.236.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.236.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.236.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.236.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.236.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

## 6.237 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1347).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.237.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1347).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.237.2 Constructor & Destructor Documentation

**6.237.2.1** `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.237.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.237.3 Member Function Documentation

**6.237.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.237.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.237.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.237.3.4** virtual void `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw ( `decaf::io::IOException` ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.237.3.5** virtual int `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw ( `decaf::io::IOException` ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.237.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.237.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h

## 6.238 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1351).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.238.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionIdMarshaller** (p.1351).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.238.2 Constructor & Destructor Documentation

**6.238.2.1** `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

**6.238.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

## 6.238.3 Member Function Documentation

**6.238.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.238.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.238.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.238.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.238.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.238.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.238.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

## 6.239 activemq::commands::ConnectionInfo Class Reference

#include <src/main/activemq/commands/ConnectionInfo.h> Inheritance diagram for activemq::commands::ConnectionInfo:

### Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ConnectionId > & getConnectionId** () const
- virtual **Pointer< ConnectionId > & getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer< ConnectionId > &connectionId**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)

- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool **clientMaster**)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID \_ CONNECTIONINFO** = 3

## Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**
- bool **faultTolerant**

## 6.239.1 Constructor & Destructor Documentation

**6.239.1.1** **activemq::commands::ConnectionInfo::ConnectionInfo** ()

**6.239.1.2** **virtual activemq::commands::ConnectionInfo::~~ConnectionInfo** ()  
[virtual]

## 6.239.2 Member Function Documentation

**6.239.2.1** **virtual ConnectionInfo\* activemq::commands::ConnectionInfo::cloneDataStructure** ()  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

### 6.239.2.2 virtual void activemq::commands::ConnectionInfo::copyDataStructure (const DataStructure \* *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

### 6.239.2.3 Pointer<RemoveInfo> activemq::commands::ConnectionInfo::createRemoveCommand () const

### 6.239.2.4 virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure \* *value*) const [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

### 6.239.2.5 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () [virtual]

### 6.239.2.6 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () const [virtual]

### 6.239.2.7 virtual std::string& activemq::commands::ConnectionInfo::getClientId () [virtual]

### 6.239.2.8 virtual const std::string& activemq::commands::ConnectionInfo::getClientId () const [virtual]

### 6.239.2.9 virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () [virtual]

### 6.239.2.10 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () const [virtual]

### 6.239.2.11 virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSet** (p. 1660) type copy.

Implements **activemq::commands::DataSet** (p. 1662).

**6.239.2.12** `virtual std::string& activemq::commands::ConnectionInfo::getPassword()  
() [virtual]`

**6.239.2.13** `virtual const std::string& activemq::commands::ConnectionInfo::getPassword()  
() const [virtual]`

**6.239.2.14** `virtual std::string& activemq::commands::ConnectionInfo::getUserName()  
() [virtual]`

**6.239.2.15** `virtual const std::string& activemq::commands::ConnectionInfo::getUserName()  
() const [virtual]`

**6.239.2.16** `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector()  
() const [virtual]`

**6.239.2.17** `virtual bool activemq::commands::ConnectionInfo::isClientMaster()  
() const [virtual]`

**6.239.2.18** `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo()  
() const [inline, virtual]`

**Returns:**

an answer of true to the **isConnectionInfo()** (p. 1358) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 761).

- 6.239.2.19 virtual bool activemq::commands::ConnectionInfo::isFaultTolerant ()  
const [virtual]
- 6.239.2.20 virtual bool activemq::commands::ConnectionInfo::isManageable ()  
const [virtual]
- 6.239.2.21 virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector (bool  
*brokerMasterConnector*) [virtual]
- 6.239.2.22 virtual void activemq::commands::ConnectionInfo::setBrokerPath  
(const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*)  
[virtual]
- 6.239.2.23 virtual void activemq::commands::ConnectionInfo::setClientId (const  
std::string & *clientId*) [virtual]
- 6.239.2.24 virtual void activemq::commands::ConnectionInfo::setClientMaster  
(bool *clientMaster*) [virtual]
- 6.239.2.25 virtual void activemq::commands::ConnectionInfo::setConnectionId  
(const Pointer< ConnectionId > & *connectionId*) [virtual]
- 6.239.2.26 virtual void activemq::commands::ConnectionInfo::setFaultTolerant  
(bool *faultTolerant*) [virtual]
- 6.239.2.27 virtual void activemq::commands::ConnectionInfo::setManageable (bool  
*manageable*) [virtual]
- 6.239.2.28 virtual void activemq::commands::ConnectionInfo::setPassword (const  
std::string & *password*) [virtual]
- 6.239.2.29 virtual void activemq::commands::ConnectionInfo::setUserName (const  
std::string & *userName*) [virtual]
- 6.239.2.30 virtual std::string activemq::commands::ConnectionInfo::toString ()  
const [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.239.2.31 virtual Pointer<Command> activemq::commands::ConnectionInfo::visit  
(activemq::state::CommandVisitor \* *visitor*) throw (  
exceptions::ActiveMQException ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

**6.239.3 Field Documentation**

- 6.239.3.1** `bool activemq::commands::ConnectionInfo::brokerMasterConnector`  
[protected]
- 6.239.3.2** `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.239.3.3** `std::string activemq::commands::ConnectionInfo::clientId` [protected]
- 6.239.3.4** `bool activemq::commands::ConnectionInfo::clientMaster` [protected]
- 6.239.3.5** `Pointer<ConnectionId> ac-`  
`tivemq::commands::ConnectionInfo::connectionId`  
[protected]
- 6.239.3.6** `bool activemq::commands::ConnectionInfo::faultTolerant` [protected]
- 6.239.3.7** `const unsigned char activemq::commands::ConnectionInfo::ID _-`  
`CONNECTIONINFO = 3` [static]
- 6.239.3.8** `bool activemq::commands::ConnectionInfo::manageable` [protected]
- 6.239.3.9** `std::string activemq::commands::ConnectionInfo::password` [protected]
- 6.239.3.10** `std::string activemq::commands::ConnectionInfo::userName`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`



## 6.240 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1361).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.240.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1361).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.240.2 Constructor & Destructor Documentation

**6.240.2.1** `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoMarshaller()` [inline]

**6.240.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller()` [inline, virtual]

## 6.240.3 Member Function Documentation

**6.240.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.240.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.240.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.240.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.240.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.240.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.240.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h

## 6.241 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1365).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.241.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1365).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.241.2 Constructor & Destructor Documentation

**6.241.2.1** `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoM  
( ) [inline]`

**6.241.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfo  
( ) [inline, virtual]`

## 6.241.3 Member Function Documentation

**6.241.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.241.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureT  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.241.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.241.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.241.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.241.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.241.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h



## 6.242 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1369).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.242.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1369).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.242.2 Constructor & Destructor Documentation

**6.242.2.1** `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoM  
( ) [inline]`

**6.242.2.2** `virtual  
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfo  
( ) [inline, virtual]`

## 6.242.3 Member Function Documentation

**6.242.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.242.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureT  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.242.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.242.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.242.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.242.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.242.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h

## 6.243 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1373).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.243.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1373).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.243.2 Constructor & Destructor Documentation

**6.243.2.1** `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoMarshaller()` [inline]

**6.243.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller()` [inline, virtual]

## 6.243.3 Member Function Documentation

**6.243.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.243.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.243.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.243.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.243.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.243.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.243.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h



## 6.244 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1377).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.244.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1377).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.244.2 Constructor & Destructor Documentation

**6.244.2.1** `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoM  
( ) [inline]`

**6.244.2.2** `virtual  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfo  
( ) [inline, virtual]`

## 6.244.3 Member Function Documentation

**6.244.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.244.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureT  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.244.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.244.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.244.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.244.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.244.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h

## 6.245 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1381).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.245.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1381).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.245.2 Constructor & Destructor Documentation

**6.245.2.1** `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::ConnectionInfoM  
( ) [inline]`

**6.245.2.2** `virtual  
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::~~ConnectionInfo  
( ) [inline, virtual]`

## 6.245.3 Member Function Documentation

**6.245.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.245.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::getDataStructureT  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.245.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.245.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.245.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.245.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.245.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h



## 6.246 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p.1385) object provides information describing the **Connection** (p.1262) object.

#include <src/main/cms/ConnectionMetaData.h> Inheritance diagram for cms::ConnectionMetaData:

### Public Member Functions

- virtual **~ConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS minor version number.*
- virtual std::string **getCMSProviderName** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider major version number.*
- virtual int **getProviderMinorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0 throw ( cms::CMSEException )  
*Gets an Vector of the CMSX property names.*

### 6.246.1 Detailed Description

A **ConnectionMetaData** (p.1385) object provides information describing the **Connection** (p.1262) object.

Since:

1.3

## 6.246.2 Constructor & Destructor Documentation

**6.246.2.1** `virtual cms::ConnectionMetaData::~~ConnectionMetaData () [inline, virtual]`

## 6.246.3 Member Function Documentation

**6.246.3.1** `virtual int cms::ConnectionMetaData::getCMSMajorVersion () const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS major version number.

### Returns:

the CMS API major version number

### Exceptions:

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 304).

**6.246.3.2** `virtual int cms::ConnectionMetaData::getCMSMinorVersion () const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS minor version number.

### Returns:

the CMS API minor version number

### Exceptions:

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 304).

**6.246.3.3** `virtual std::string cms::ConnectionMetaData::getCMSProviderName () const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS provider name.

### Returns:

the CMS provider name

### Exceptions:

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 304).

**6.246.3.4** `virtual std::string cms::ConnectionMetaData::getCMSVersion () const  
throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS API version.

**Returns:**

the CMS API Version in String form.

**Exceptions:**

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 305).

**6.246.3.5** `virtual std::vector<std::string>  
cms::ConnectionMetaData::getCMSXPropertyNames ()  
const throw ( cms::CMSEException ) [pure virtual]`

Gets an Vector of the CMSX property names.

**Returns:**

an Vector of CMSX property names

**Exceptions:**

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 305).

**6.246.3.6** `virtual int cms::ConnectionMetaData::getProviderMajorVersion () const  
throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS provider major version number.

**Returns:**

the CMS provider major version number

**Exceptions:**

*CMSEException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 305).

**6.246.3.7** `virtual int cms::ConnectionMetaData::getProviderMinorVersion () const  
throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS provider minor version number.

**Returns:**

the CMS provider minor version number

**Exceptions:**

*CMSException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 306).

**6.246.3.8** `virtual std::string cms::ConnectionMetaData::getProviderVersion ()  
const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider version.

**Returns:**

the CMS provider version

**Exceptions:**

*CMSException* (p. 1160) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 306).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

## 6.247 activemq::state::ConnectionState Class Reference

```
#include <src/main/activemq/state/ConnectionState.h>
```

### Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- std::string **toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- std::vector< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- std::vector< **Pointer**< **SessionState** > > **getSessionStates** () const
- **StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > **getRecoveringPullConsumers** ()
- void **setConnectionInterruptProcessingComplete** (bool connectionInterruptProcessingComplete)
- bool **isConnectionInterruptProcessingComplete** ()

## 6.247.1 Constructor & Destructor Documentation

- 6.247.1.1 `activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & info)`
- 6.247.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ()` [virtual]

## 6.247.2 Member Function Documentation

- 6.247.2.1 `void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & info)` [inline]
- 6.247.2.2 `void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & info)` [inline]
- 6.247.2.3 `void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & id)` [inline]
- 6.247.2.4 `void activemq::state::ConnectionState::checkShutdown () const`
- 6.247.2.5 `const Pointer<commands::ConnectionInfo>& activemq::state::ConnectionState::getInfo () const` [inline]
- 6.247.2.6 `StlMap< Pointer<ConsumerId>, Pointer<ConsumerInfo>, ConsumerId::COMPARATOR > & activemq::state::ConnectionState::getRecoveringPullConsumers ()` [inline]
- 6.247.2.7 `const Pointer<SessionState>& activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & id) const` [inline]
- 6.247.2.8 `std::vector< Pointer<SessionState> > & activemq::state::ConnectionState::getSessionStates () const` [inline]
- 6.247.2.9 `const StlList< Pointer<DestinationInfo> >& activemq::state::ConnectionState::getTempDesinations () const` [inline]
- 6.247.2.10 `const Pointer<TransactionState>& activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & id) const` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.11 `std::vector< Pointer<TransactionState> > activemq::state::ConnectionState::getTransactionStates () const` [inline]
- 6.247.2.12 `bool activemq::state::ConnectionState::isConnectionInterruptProcessingComplete ()` [inline]
- 6.247.2.13 `Pointer<SessionState> activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & id)` [inline]
- 6.247.2.14 `void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & destination)` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.15 `Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & id)` [inline]
- 6.247.2.16 `void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & info)`
- 6.247.2.17 `void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete (bool connectionInterruptProcessingComplete)` [inline]
- 6.247.2.18 `void activemq::state::ConnectionState::shutdown ()`
- 6.247.2.19 `std::string activemq::state::ConnectionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

## 6.248 activemq::state::ConnectionStateTracker Class Reference

#include <src/main/activemq/state/ConnectionStateTracker.h> Inheritance diagram for activemq::state::ConnectionStateTracker:

### Public Member Functions

- **ConnectionStateTracker** ()
- virtual **~ConnectionStateTracker** ()
- **Pointer< Tracked > track** (const **Pointer< Command >** &command) throw ( default::io::IOException )
- void **trackBack** (const **Pointer< Command >** &command)
- void **restore** (const **Pointer< transport::Transport >** &transport) throw ( default::io::IOException )
- void **connectionInterruptProcessingComplete** (transport::Transport \*transport, const **Pointer< ConnectionId >** &connectionId)
- void **transportInterrupted** ()
- virtual **Pointer< Command > processDestinationInfo** (DestinationInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveDestination** (DestinationInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processProducerInfo** (ProducerInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveProducer** (ProducerId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processConsumerInfo** (ConsumerInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveConsumer** (ConsumerId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processSessionInfo** (SessionInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveSession** (SessionId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processConnectionInfo** (ConnectionInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveConnection** (ConnectionId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processMessage** (Message \*message) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processMessageAck** (MessageAck \*ack) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processBeginTransaction** (TransactionInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processPrepareTransaction** (TransactionInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processCommitTransactionOnePhase** (TransactionInfo \*info) throw ( exceptions::ActiveMQException )



- virtual **Pointer< Command > processCommitTransactionTwoPhase** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

## Friends

- class **RemoveTransactionAction**

## 6.248.1 Constructor & Destructor Documentation

6.248.1.1 `activemq::state::ConnectionStateTracker::ConnectionStateTracker ()`

6.248.1.2 `virtual  
activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ()  
[virtual]`

## 6.248.2 Member Function Documentation

6.248.2.1 `void ac-  
tivismq::state::ConnectionStateTracker::connectionInterruptProcessingComplete  
(transport::Transport * transport, const Pointer< ConnectionId > &  
connectionId)`

6.248.2.2 `int activismq::state::ConnectionStateTracker::getMaxCacheSize () const  
[inline]`

6.248.2.3 `bool activismq::state::ConnectionStateTracker::isRestoreConsumers ()  
const [inline]`

6.248.2.4 `bool activismq::state::ConnectionStateTracker::isRestoreProducers ()  
const [inline]`

6.248.2.5 `bool activismq::state::ConnectionStateTracker::isRestoreSessions () const  
[inline]`

6.248.2.6 `bool activismq::state::ConnectionStateTracker::isRestoreTransaction ()  
const [inline]`

6.248.2.7 `bool activismq::state::ConnectionStateTracker::isTrackMessages () const  
[inline]`

6.248.2.8 `bool ac-  
tivismq::state::ConnectionStateTracker::isTrackTransactionProducers ()  
const [inline]`

6.248.2.9 `bool activismq::state::ConnectionStateTracker::isTrackTransactions ()  
const [inline]`

6.248.2.10 `virtual Pointer<Command> ac-  
tivismq::state::ConnectionStateTracker::processBeginTransaction  
(TransactionInfo * info) throw ( exceptions::ActiveMQException )  
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1202).

6.248.2.11 `virtual Pointer<Command> ac-  
tivismq::state::ConnectionStateTracker::processCommitTransactionOnePhase  
(TransactionInfo * info) throw ( exceptions::ActiveMQException )  
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1202).

**6.248.2.12** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase (TransactionInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1202).

**6.248.2.13** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1203).

**6.248.2.14** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1203).

**6.248.2.15** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1203).

**6.248.2.16** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1203).

**6.248.2.17** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message \* *message*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1204).

**6.248.2.18** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck \* *ack*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements activemq::state::CommandVisitor (p. 1204).

**6.248.2.19** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * info) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1204).

**6.248.2.20** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * info) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1205).

**6.248.2.21** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * id) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1205).

**6.248.2.22** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * id) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1205).

**6.248.2.23** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * info) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1205).

**6.248.2.24** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * id) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1205).

**6.248.2.25** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * id) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1206).

**6.248.2.26** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1206).

**6.248.2.27** virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo \* *info*) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1206).

- 6.248.2.28 `void activemq::state::ConnectionStateTracker::restore (const Pointer< transport::Transport > & transport) throw ( decaf::io::IOException )`
- 6.248.2.29 `void activemq::state::ConnectionStateTracker::setMaxCacheSize (int maxCacheSize) [inline]`
- 6.248.2.30 `void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool restoreConsumers) [inline]`
- 6.248.2.31 `void activemq::state::ConnectionStateTracker::setRestoreProducers (bool restoreProducers) [inline]`
- 6.248.2.32 `void activemq::state::ConnectionStateTracker::setRestoreSessions (bool restoreSessions) [inline]`
- 6.248.2.33 `void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool restoreTransaction) [inline]`
- 6.248.2.34 `void activemq::state::ConnectionStateTracker::setTrackMessages (bool trackMessages) [inline]`
- 6.248.2.35 `void activemq::state::ConnectionStateTracker::setTrackTransactionProducers (bool trackTransactionProducers) [inline]`
- 6.248.2.36 `void activemq::state::ConnectionStateTracker::setTrackTransactions (bool trackTransactions) [inline]`
- 6.248.2.37 `Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & command) throw ( decaf::io::IOException )`
- 6.248.2.38 `void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & command)`
- 6.248.2.39 `void activemq::state::ConnectionStateTracker::transportInterrupted ()`

### 6.248.3 Friends And Related Function Documentation

- 6.248.3.1 `friend class RemoveTransactionAction [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionStateTracker.h`

## 6.249 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 1978) publishes log records to System.err.

#include <src/main/decaf/util/logging/ConsoleHandler.h> Inheritance diagram for decaf::util::logging::ConsoleHandler:

### Public Member Functions

- **ConsoleHandler** ()
- virtual **~ConsoleHandler** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Close the current output stream.*
- virtual void **publish** (const **LogRecord** &record)  
*Publish the Log Record to this **Handler** (p. 1978).*

### 6.249.1 Detailed Description

This **Handler** (p. 1978) publishes log records to System.err. By default the **SimpleFormatter** (p. 3500) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 1399) is initialized using the following **LogManager** (p. 2406) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

ConsoleHandler.level specifies the default level for the **Handler** (p. 1978) (defaults to **Level.INFO** (p. 2336)). ConsoleHandler.filter specifies the name of a **Filter** (p. 1893) class to use (defaults to no **Filter** (p. 1893)). ConsoleHandler.formatter specifies the name of a **Formatter** (p. 1964) class to use (defaults to **SimpleFormatter** (p. 3500)).

Since:

1.0

### 6.249.2 Constructor & Destructor Documentation

6.249.2.1 decaf::util::logging::ConsoleHandler::ConsoleHandler ()

6.249.2.2 virtual decaf::util::logging::ConsoleHandler::~~ConsoleHandler ()  
[inline, virtual]

### 6.249.3 Member Function Documentation

6.249.3.1 virtual void decaf::util::logging::ConsoleHandler::close () throw ( decaf::io::IOException ) [virtual]

Close the current output stream. Override the **StreamHandler** (p. 3649) close to flush the Std Err stream but doesn't close.

**Exceptions:*****IOException***

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3650).

**6.249.3.2 virtual void decaf::util::logging::ConsoleHandler::publish (const LogRecord & *record*) [virtual]**

Publish the Log Record to this **Handler** (p. 1978).

**Parameters:**

*record* The LogRecord (p. 2413) to Publish

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3651).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ConsoleHandler.h`



## 6.250 activemq::commands::ConsumerControl Class Reference

#include <src/main/activemq/commands/ConsumerControl.h> Inheritance diagram for activemq::commands::ConsumerControl:

### Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerControl** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const
- virtual void **setStart** (bool start)
- virtual bool **isStop** () const
- virtual void **setStop** (bool stop)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_CONSUMERCONTROL** = 17

## Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- bool **close**
- **Pointer**< **ConsumerId** > **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

## 6.250.1 Constructor & Destructor Documentation

**6.250.1.1** **activemq::commands::ConsumerControl::ConsumerControl** ()

**6.250.1.2** **virtual activemq::commands::ConsumerControl::~~ConsumerControl** ()  
[virtual]

## 6.250.2 Member Function Documentation

**6.250.2.1** **virtual ConsumerControl\* activemq::commands::ConsumerControl::cloneDataStructure**  
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.250.2.2** **virtual void activemq::commands::ConsumerControl::copyDataStructure**  
(const **DataStructure** \* *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.250.2.3** **virtual bool activemq::commands::ConsumerControl::equals** (const  
**DataStructure** \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.250.2.4** virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ()  
[virtual]

**6.250.2.5** virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ()  
const [virtual]

**6.250.2.6** virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType () const  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.250.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ()`  
[virtual]
- 6.250.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination () const`  
[virtual]
- 6.250.2.9 `virtual int activemq::commands::ConsumerControl::getPrefetch () const`  
[virtual]
- 6.250.2.10 `virtual bool activemq::commands::ConsumerControl::isClose () const`  
[virtual]
- 6.250.2.11 `virtual bool activemq::commands::ConsumerControl::isFlush () const`  
[virtual]
- 6.250.2.12 `virtual bool activemq::commands::ConsumerControl::isStart () const`  
[virtual]
- 6.250.2.13 `virtual bool activemq::commands::ConsumerControl::isStop () const`  
[virtual]
- 6.250.2.14 `virtual void activemq::commands::ConsumerControl::setClose (bool close)` [virtual]
- 6.250.2.15 `virtual void activemq::commands::ConsumerControl::setConsumerId (const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.250.2.16 `virtual void activemq::commands::ConsumerControl::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.250.2.17 `virtual void activemq::commands::ConsumerControl::setFlush (bool flush)` [virtual]
- 6.250.2.18 `virtual void activemq::commands::ConsumerControl::setPrefetch (int prefetch)` [virtual]
- 6.250.2.19 `virtual void activemq::commands::ConsumerControl::setStart (bool start)` [virtual]
- 6.250.2.20 `virtual void activemq::commands::ConsumerControl::setStop (bool stop)` [virtual]
- 6.250.2.21 `virtual std::string activemq::commands::ConsumerControl::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

**6.250.2.22** `virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1198).

### 6.250.3 Field Documentation

**6.250.3.1** `bool activemq::commands::ConsumerControl::close [protected]`

**6.250.3.2** `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId [protected]`

**6.250.3.3** `Pointer<ActiveMQDestination> activemq::commands::ConsumerControl::destination [protected]`

**6.250.3.4** `bool activemq::commands::ConsumerControl::flush [protected]`

**6.250.3.5** `const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17 [static]`

**6.250.3.6** `int activemq::commands::ConsumerControl::prefetch [protected]`

**6.250.3.7** `bool activemq::commands::ConsumerControl::start [protected]`

**6.250.3.8** `bool activemq::commands::ConsumerControl::stop [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

## 6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1406).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.251.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1406).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.251.2 Constructor & Destructor Documentation

6.251.2.1 **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller()** [inline]

6.251.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller()** [inline, virtual]

## 6.251.3 Member Function Documentation

6.251.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject()** const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1611).

6.251.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType()** const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1617).

6.251.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal(OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut) throw (decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.251.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.251.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).



**6.251.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.251.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h

## 6.252 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1410).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.252.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1410).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.252.2 Constructor & Destructor Documentation

**6.252.2.1** `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

**6.252.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

## 6.252.3 Member Function Documentation

**6.252.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.252.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.252.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.252.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.252.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.252.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.252.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h

## 6.253 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1414).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.253.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1414).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.253.2 Constructor & Destructor Documentation

**6.253.2.1** `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

**6.253.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

## 6.253.3 Member Function Documentation

**6.253.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.253.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.253.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.253.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.253.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).



**6.253.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.253.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h

## 6.254 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1418).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.254.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1418).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.254.2 Constructor & Destructor Documentation

6.254.2.1 **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller()** [inline]

6.254.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller()** [inline, virtual]

## 6.254.3 Member Function Documentation

6.254.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject()** const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1611).

6.254.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType()** const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1617).

6.254.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal(OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut) throw (decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.254.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.254.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.254.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.254.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h

## 6.255 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1422).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.255.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1422).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.255.2 Constructor & Destructor Documentation

**6.255.2.1** `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

**6.255.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

## 6.255.3 Member Function Documentation

**6.255.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.255.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.255.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.255.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.255.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).



**6.255.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.255.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h

## 6.256 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1426).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.256.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1426).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.256.2 Constructor & Destructor Documentation

6.256.2.1 **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::ConsumerControlMarshaller()** [inline]

6.256.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::~~ConsumerControlMarshaller()** [inline, virtual]

## 6.256.3 Member Function Documentation

6.256.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::createObject()** const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1611).

6.256.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::getDataStructureType()** const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1617).

6.256.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseMarshal(OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut) throw (decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.256.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.256.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.256.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.256.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h

## 6.257 activemq::commands::ConsumerId Class Reference

#include <src/main/activemq/commands/ConsumerId.h> Inheritance diagram for activemq::commands::ConsumerId:

### Public Types

- typedef decaf::lang::PointerComparator< ConsumerId > COMPARATOR

### Public Member Functions

- ConsumerId ()
- ConsumerId (const ConsumerId &other)
- ConsumerId (const SessionId &sessionId, long long consumerId)
- virtual ~ConsumerId ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual ConsumerId \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const DataStructure \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this DataStructure (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const DataStructure \*value) const  
*Compares the DataStructure (p. 1660) passed in to this one, and returns if they are equivalent.*
- const Pointer< SessionId > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const ConsumerId &value) const
- virtual bool **equals** (const ConsumerId &value) const
- virtual bool **operator==** (const ConsumerId &value) const
- virtual bool **operator<** (const ConsumerId &value) const
- ConsumerId & **operator=** (const ConsumerId &other)

## Static Public Attributes

- static const unsigned char **ID\_ CONSUMERID** = 122

## Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

### 6.257.1 Member Typedef Documentation

- 6.257.1.1**    `typedef decaf::lang::PointerComparator<ConsumerId>  
activemq::commands::ConsumerId::COMPARATOR`

### 6.257.2 Constructor & Destructor Documentation

- 6.257.2.1**    `activemq::commands::ConsumerId::ConsumerId ()`
- 6.257.2.2**    `activemq::commands::ConsumerId::ConsumerId (const ConsumerId &  
other)`
- 6.257.2.3**    `activemq::commands::ConsumerId::ConsumerId (const SessionId &  
sessionId, long long consumerId)`
- 6.257.2.4**    `virtual activemq::commands::ConsumerId::~~ConsumerId ()` [virtual]

### 6.257.3 Member Function Documentation

- 6.257.3.1**    `virtual ConsumerId* ac-  
tivemq::commands::ConsumerId::cloneDataStructure ()  
const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p.1660).

- 6.257.3.2**    `virtual int activemq::commands::ConsumerId::compareTo (const  
ConsumerId & value) const` [virtual]

- 6.257.3.3**    `virtual void activemq::commands::ConsumerId::copyDataStructure  
(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

**6.257.3.4** `virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & value) const` [virtual]

**6.257.3.5** `virtual bool activemq::commands::ConsumerId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

**6.257.3.6** `virtual std::string& activemq::commands::ConsumerId::getConnectionId ()` [virtual]

**6.257.3.7** `virtual const std::string& activemq::commands::ConsumerId::getConnectionId () const` [virtual]

**6.257.3.8** `virtual unsigned char activemq::commands::ConsumerId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).



- 6.257.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId () const`
- 6.257.3.10 `virtual long long activemq::commands::ConsumerId::getSessionId () const [virtual]`
- 6.257.3.11 `virtual long long activemq::commands::ConsumerId::getValue () const [virtual]`
- 6.257.3.12 `virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & value) const [virtual]`
- 6.257.3.13 `ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & other)`
- 6.257.3.14 `virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & value) const [virtual]`
- 6.257.3.15 `virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.257.3.16 `virtual void activemq::commands::ConsumerId::setSessionId (long long sessionId) [virtual]`
- 6.257.3.17 `virtual void activemq::commands::ConsumerId::setValue (long long value) [virtual]`
- 6.257.3.18 `virtual std::string activemq::commands::ConsumerId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 833).

## 6.257.4 Field Documentation

- 6.257.4.1 `std::string activemq::commands::ConsumerId::connectionId [protected]`
- 6.257.4.2 `const unsigned char activemq::commands::ConsumerId::ID _ - CONSUMERID = 122 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

**6.257.4.3**   `long long activemq::commands::ConsumerId::sessionId`   [protected]

**6.257.4.4**   `long long activemq::commands::ConsumerId::value`   [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerId.h`

## 6.258 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1435).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.258.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1435).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.258.2 Constructor & Destructor Documentation

**6.258.2.1** `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.258.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.258.3 Member Function Documentation

**6.258.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.258.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.258.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.258.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.258.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.258.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.258.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h

## 6.259 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1439).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.259.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1439).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.259.2 Constructor & Destructor Documentation

**6.259.2.1** `activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.259.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.259.3 Member Function Documentation

**6.259.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.259.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.259.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.259.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.259.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.259.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.259.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h

## 6.260 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1443).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.260.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1443).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.260.2 Constructor & Destructor Documentation

**6.260.2.1** `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.260.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.260.3 Member Function Documentation

**6.260.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.260.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.260.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.260.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.260.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.260.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.260.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h

## 6.261 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1447).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.261.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1447).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.261.2 Constructor & Destructor Documentation

**6.261.2.1** `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.261.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.261.3 Member Function Documentation

**6.261.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.261.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.261.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.261.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.261.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.261.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.261.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h

## 6.262 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1451).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.262.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1451).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.262.2 Constructor & Destructor Documentation

**6.262.2.1** `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.262.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.262.3 Member Function Documentation

**6.262.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.262.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.262.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.262.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.262.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.262.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.262.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h

## 6.263 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1455).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.263.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerIdMarshaller** (p.1455).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.263.2 Constructor & Destructor Documentation

**6.263.2.1** `activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

**6.263.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

## 6.263.3 Member Function Documentation

**6.263.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.263.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.263.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.263.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.263.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.263.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.263.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h

## 6.264 activemq::commands::ConsumerInfo Class Reference

#include <src/main/activemq/commands/ConsumerInfo.h> Inheritance diagram for activemq::commands::ConsumerInfo:

### Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)

- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool **noLocal**)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool **retroactive**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **brokerPath**)
- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > & **additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID \_CONSUMERINFO** = 5

## Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**

- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- Pointer< BooleanExpression > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< decaf::lang::Pointer< ConsumerId > > **networkConsumerPath**

### 6.264.1 Constructor & Destructor Documentation

- 6.264.1.1** `activemq::commands::ConsumerInfo::ConsumerInfo ()`
- 6.264.1.2** `virtual activemq::commands::ConsumerInfo::~~ConsumerInfo ()`  
[virtual]

### 6.264.2 Member Function Documentation

- 6.264.2.1** `virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

- 6.264.2.2** `virtual void activemq::commands::ConsumerInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.264.2.3** `Pointer<RemoveInfo> activemq::commands::ConsumerInfo::createRemoveCommand ()`  
`const`
- 6.264.2.4** `virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.264.2.5 `virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ()` [virtual]
- 6.264.2.6 `virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () const` [virtual]
- 6.264.2.7 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ()` [virtual]
- 6.264.2.8 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () const` [virtual]
- 6.264.2.9 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ()` [virtual]
- 6.264.2.10 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () const` [virtual]
- 6.264.2.11 `virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.264.2.12 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ()` [virtual]
- 6.264.2.13 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () const` [virtual]
- 6.264.2.14 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit () const` [virtual]
- 6.264.2.15 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ()` [virtual]
- 6.264.2.16 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath () const` [virtual]
- 6.264.2.17 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize () const` [virtual]
- 6.264.2.18 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority () const` [virtual]
- 6.264.2.19 `virtual std::string& activemq::commands::ConsumerInfo::getSelector ()` [virtual]
- 6.264.2.20 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector () const` [virtual]
- 6.264.2.21 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ()` [virtual]
- 6.264.2.22 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const` [virtual]
- 6.264.2.23 `virtual bool activemq::commands::ConsumerInfo::isBrowser () const` [virtual]
- 6.264.2.24 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const` [inline, virtual]

**Returns:**

an answer of true to the `isConsumerInfo()` (p. 1463) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 761).

- 6.264.2.25 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync ()  
const [virtual]
- 6.264.2.26 virtual bool activemq::commands::ConsumerInfo::isExclusive () const  
[virtual]
- 6.264.2.27 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription  
() const [virtual]
- 6.264.2.28 virtual bool activemq::commands::ConsumerInfo::isNoLocal () const  
[virtual]
- 6.264.2.29 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks ()  
const [virtual]
- 6.264.2.30 virtual bool ac-  
tivismq::commands::ConsumerInfo::isOptimizedAcknowledge () const  
[virtual]
- 6.264.2.31 virtual bool activemq::commands::ConsumerInfo::isRetroactive () const  
[virtual]
- 6.264.2.32 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate  
(const Pointer< BooleanExpression > & *additionalPredicate*) [virtual]
- 6.264.2.33 virtual void activemq::commands::ConsumerInfo::setBrokerPath  
(const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*)  
[virtual]
- 6.264.2.34 virtual void activemq::commands::ConsumerInfo::setBrowser (bool  
*browser*) [virtual]
- 6.264.2.35 virtual void activemq::commands::ConsumerInfo::setConsumerId (const  
Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.264.2.36 virtual void activemq::commands::ConsumerInfo::setDestination (const  
Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.264.2.37 virtual void activemq::commands::ConsumerInfo::setDispatchAsync  
(bool *dispatchAsync*) [virtual]
- 6.264.2.38 virtual void activemq::commands::ConsumerInfo::setExclusive (bool  
*exclusive*) [virtual]
- 6.264.2.39 virtual void ac-  
tivismq::commands::ConsumerInfo::setMaximumPendingMessageLimit  
(int *maximumPendingMessageLimit*) [virtual]
- 6.264.2.40 virtual void ac-  
tivismq::commands::ConsumerInfo::setNetworkConsumerPath  
(const std::vector< decaf::lang::Pointer< ConsumerId > > &  
*networkConsumerPath*) [virtual]
- 6.264.2.41 virtual void ac-  
tivismq::commands::ConsumerInfo::setNetworkSubscription (bool  
*networkSubscription*) [virtual]
- 6.264.2.42 virtual void activemq::commands::ConsumerInfo::setNoLocal (bool  
*noLocal*) [virtual]
- 6.264.2.43 virtual void activemq::commands::ConsumerInfo::setNoRangeAcks  
(bool *noRangeAcks*) [virtual]



**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.264.2.51** `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit  
(activemq::state::CommandVisitor * visitor) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.264.3 Field Documentation

- 6.264.3.1 `Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate`  
[protected]
- 6.264.3.2 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath` [protected]
- 6.264.3.3 `bool activemq::commands::ConsumerInfo::browser` [protected]
- 6.264.3.4 `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId`  
[protected]
- 6.264.3.5 `Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination`  
[protected]
- 6.264.3.6 `bool activemq::commands::ConsumerInfo::dispatchAsync` [protected]
- 6.264.3.7 `bool activemq::commands::ConsumerInfo::exclusive` [protected]
- 6.264.3.8 `const unsigned char activemq::commands::ConsumerInfo::ID_ - CONSUMERINFO = 5` [static]
- 6.264.3.9 `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit`  
[protected]
- 6.264.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath`  
[protected]
- 6.264.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription`  
[protected]
- 6.264.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.264.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks` [protected]
- 6.264.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge`  
[protected]
- 6.264.3.15 `int activemq::commands::ConsumerInfo::prefetchSize` [protected]
- 6.264.3.16 `unsigned char activemq::commands::ConsumerInfo::priority`  
[protected]
- 6.264.3.17 `bool activemq::commands::ConsumerInfo::retroactive` [protected]
- 6.264.3.18 `std::string activemq::commands::ConsumerInfo::selector` [protected]
- 6.264.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName`  
[protected]

The documentation for this class was generated from the following file:

Generated on Wed Jul 25 23:57:04 2012 for activemq-cpp-3.2.5 by Doxygen

- `src/main/activemq/commands/ConsumerInfo.h`

## 6.265 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1468).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.265.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1468).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.265.2 Constructor & Destructor Documentation

**6.265.2.1** `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

**6.265.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

## 6.265.3 Member Function Documentation

**6.265.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.265.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.265.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.265.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.265.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.265.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.265.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

## 6.266 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1472).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.266.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1472).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.266.2 Constructor & Destructor Documentation

**6.266.2.1** `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMar  
( ) [inline]`

**6.266.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMar  
( ) [inline, virtual]`

## 6.266.3 Member Function Documentation

**6.266.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.266.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.266.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.266.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.266.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.266.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.266.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h

## 6.267 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1476).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.267.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1476).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.267.2 Constructor & Destructor Documentation

**6.267.2.1** `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMar  
( ) [inline]`

**6.267.2.2** `virtual  
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMar  
( ) [inline, virtual]`

## 6.267.3 Member Function Documentation

**6.267.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.267.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.267.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.267.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.267.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.267.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.267.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

## 6.268 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1480).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.268.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1480).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.268.2 Constructor & Destructor Documentation

**6.268.2.1** `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` `[inline]`

**6.268.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` `[inline, virtual]`

## 6.268.3 Member Function Documentation

**6.268.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.268.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.268.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.268.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.268.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.268.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.268.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h

## 6.269 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1484).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.269.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1484).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.269.2 Constructor & Destructor Documentation

**6.269.2.1** `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

**6.269.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

## 6.269.3 Member Function Documentation

**6.269.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.269.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.269.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.269.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.269.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.269.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.269.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h

## 6.270 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1488).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.270.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ConsumerInfoMarshaller** (p.1488).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.270.2 Constructor & Destructor Documentation

**6.270.2.1** `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

**6.270.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

## 6.270.3 Member Function Documentation

**6.270.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.270.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.270.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.270.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.270.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.270.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.270.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

## 6.271 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

### Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual **~ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

### 6.271.1 Constructor & Destructor Documentation

**6.271.1.1** **activemq::state::ConsumerState::ConsumerState** (const **Pointer**< **ConsumerInfo** > & *info*)

**6.271.1.2** virtual **activemq::state::ConsumerState::~~ConsumerState** () [virtual]

### 6.271.2 Member Function Documentation

**6.271.2.1** const **Pointer**<**ConsumerInfo**>& **activemq::state::ConsumerState::getInfo** () const [inline]

**6.271.2.2** std::string **activemq::state::ConsumerState::toString** () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

## 6.272 activemq::commands::ControlCommand Class Reference

#include <src/main/activemq/commands/ControlCommand.h> Inheritance diagram for activemq::commands::ControlCommand:

### Public Member Functions

- **ControlCommand** ()
- virtual **~ControlCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ControlCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &command)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID \_CONTROLCOMMAND** = 14

### Protected Attributes

- std::string **command**

## 6.272.1 Constructor & Destructor Documentation

**6.272.1.1** `activemq::commands::ControlCommand::ControlCommand ()`

**6.272.1.2** `virtual activemq::commands::ControlCommand::~~ControlCommand ()`  
[virtual]

## 6.272.2 Member Function Documentation

**6.272.2.1** `virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.272.2.2** `virtual void activemq::commands::ControlCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.272.2.3** `virtual bool activemq::commands::ControlCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.272.2.4** `virtual std::string& activemq::commands::ControlCommand::getCommand ()`  
[virtual]
- 6.272.2.5** `virtual const std::string& activemq::commands::ControlCommand::getCommand ()`  
`const` [virtual]
- 6.272.2.6** `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType ()` `const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSetructure** (p. 1660) type copy.

Implements **activemq::commands::DataSetructure** (p. 1662).

- 6.272.2.7** `virtual void activemq::commands::ControlCommand::setCommand (const std::string & command)` [virtual]
- 6.272.2.8** `virtual std::string activemq::commands::ControlCommand::toString ()`  
`const` [virtual]

Returns a string containing the information for this **DataSetructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.272.2.9** `virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * visitor)` `throw (exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.272.3 Field Documentation

**6.272.3.1** `std::string activemq::commands::ControlCommand::command`  
[protected]

**6.272.3.2** `const unsigned char activemq::commands::ControlCommand::ID_ -  
CONTROLCOMMAND = 14` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ControlCommand.h`



## 6.273 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1497).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.273.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1497).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.273.2 Constructor & Destructor Documentation

**6.273.2.1** `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.273.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.273.3 Member Function Documentation

**6.273.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.273.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.273.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 801).

**6.273.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 802).

**6.273.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 803).

**6.273.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.273.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h

## 6.274 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1501).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.274.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1501).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.274.2 Constructor & Destructor Documentation

**6.274.2.1** `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.274.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.274.3 Member Function Documentation

**6.274.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.274.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.274.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.274.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.274.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.274.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.274.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h



## 6.275 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1505).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.275.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1505).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.275.2 Constructor & Destructor Documentation

**6.275.2.1** `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.275.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.275.3 Member Function Documentation

**6.275.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.275.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.275.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.275.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.275.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.275.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.275.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h

## 6.276 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1509).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.276.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1509).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.276.2 Constructor & Destructor Documentation

**6.276.2.1** `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.276.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.276.3 Member Function Documentation

**6.276.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.276.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.276.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 780).

**6.276.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal**  
(**OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 781).

**6.276.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1**  
(**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException**  
) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 782).

**6.276.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.276.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h



## 6.277 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1513).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.277.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1513).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.277.2 Constructor & Destructor Documentation

**6.277.2.1** `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.277.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.277.3 Member Function Documentation

**6.277.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.277.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.277.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.277.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.277.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.277.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.277.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h

## 6.278 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1517).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.278.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1517).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.278.2 Constructor & Destructor Documentation

**6.278.2.1** `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::ControlComm`  
`() [inline]`

**6.278.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::~~ControlComm`  
`() [inline, virtual]`

## 6.278.3 Member Function Documentation

**6.278.3.1** `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::createObject`  
`() const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.278.3.2** `virtual unsigned char ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::getDataStructur`  
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.278.3.3** `virtual void ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseMarshal`  
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.278.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.278.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.278.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.278.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h



## 6.279 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

### Public Member Functions

- **CountDownLatch** (int count)

*Constructor.*

- virtual **~CountDownLatch** ()
- virtual void **await** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.*

- virtual bool **await** (long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*

- virtual bool **await** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*

- virtual void **countDown** ()

*Counts down the latch, releasing all waiting threads when the count hits zero.*

- virtual int **getCount** () const

*Gets the current count.*

### 6.279.1 Constructor & Destructor Documentation

#### 6.279.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int count)

Constructor.

#### Parameters:

**count** - number to count down from.

**6.279.1.2** `virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch ()`  
[virtual]

## 6.279.2 Member Function Documentation

**6.279.2.1** `virtual bool decaf::util::concurrent::CountDownLatch::await`  
(long long *timeout*, const TimeUnit & *unit*) throw (  
decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )  
[virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses. If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

\* The count reaches zero due to invocations of the **countDown()** (p.1523) method; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

### Parameters:

*timeout* - Time to wait for the count to reach zero.

*unit* - The units that the timeout specifies.

### Exceptions:

**InterruptedException** - if the current thread is interrupted while waiting.

**Exception** - if any other error occurs.

**6.279.2.2** `virtual bool decaf::util::concurrent::CountDownLatch::await` (long  
long *timeOut*) throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::Exception ) [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses. If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

\* The count reaches zero due to invocations of the **countDown()** (p.1523) method; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared. If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

**Parameters:**

*timeout* - Time in milliseconds to wait for the count to reach zero.

**Exceptions:**

*InterruptedException* - if the current thread is interrupted while waiting.

*Exception* - if any other error occurs.

**6.279.2.3** virtual void decaf::util::concurrent::CountDownLatch::await () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )  
[virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted. If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

\* The count reaches zero due to invocations of the **countDown()** (p. 1523) method; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared.

**Exceptions:**

*InterruptedException* - if the current thread is interrupted while waiting.

*Exception* - if any other error occurs.

**6.279.2.4** virtual void decaf::util::concurrent::CountDownLatch::countDown ()  
[virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

**6.279.2.5** virtual int decaf::util::concurrent::CountDownLatch::getCount () const  
[inline, virtual]

Gets the current count.

**Returns:**

int count value

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/CountDownLatch.h

## 6.280 decaf::util::zip::CRC32 Class Reference

Class that can be used to compute a CRC-32 checksum for a data stream.

```
#include <src/main/decaf/util/zip/CRC32.h>
Inheritance diagram for decaf::util::zip::CRC32:
```

### Public Member Functions

- **CRC32** ()
- virtual **~CRC32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)  
*Updates the current checksum with the specified byte value.*

### 6.280.1 Detailed Description

Class that can be used to compute a CRC-32 checksum for a data stream.

**Since:**

1.0

## 6.280.2 Constructor & Destructor Documentation

**6.280.2.1** decaf::util::zip::CRC32::CRC32 ()

**6.280.2.2** virtual decaf::util::zip::CRC32::~~CRC32 () [virtual]

## 6.280.3 Member Function Documentation

**6.280.3.1** virtual long long decaf::util::zip::CRC32::getValue () const [virtual]

### Returns:

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1142).

**6.280.3.2** virtual void decaf::util::zip::CRC32::reset () [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.280.3.3** virtual void decaf::util::zip::CRC32::update (int *byte*) [virtual]

Updates the current checksum with the specified byte value.

### Parameters:

*byte* The byte value to update the current **Checksum** (p. 1142) with (0..255).

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.280.3.4** virtual void decaf::util::zip::CRC32::update (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Updates the current checksum with the specified array of bytes.

### Parameters:

*buffer* The buffer to read the updated bytes from.

*size* The size of the passed buffer.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

### Exceptions:

**NullPointerException** if the passed buffer is NULL.

**IndexOutOfBoundsException** if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1143).

**6.280.3.5** `virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Updates the current checksum with the specified array of bytes.

**Parameters:**

*buffer* The buffer to read the updated bytes from.

*offset* The position in the buffer to start reading.

*length* The amount of data to read from the byte buffer.

**Exceptions:**

*IndexOutOfBoundsException* if  $\text{offset} + \text{length} > \text{size of the buffer}$ .

Implements `decaf::util::zip::Checksum` (p. 1143).

**6.280.3.6** `virtual void decaf::util::zip::CRC32::update (const std::vector< unsigned char > & buffer)` [virtual]

Updates the current checksum with the specified vector of bytes.

**Parameters:**

*buffer* The buffer to read the updated bytes from.

Implements `decaf::util::zip::Checksum` (p. 1144).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CRC32.h`

## 6.281 ct\_data\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- union {  
    ush freq  
    ush code  
} fc
- union {  
    ush dad  
    ush len  
} dl

### 6.281.1 Field Documentation

6.281.1.1 ush ct\_data\_s::code

6.281.1.2 ush ct\_data\_s::dad

6.281.1.3 union { ... } ct\_data\_s::dl

6.281.1.4 union { ... } ct\_data\_s::fc

6.281.1.5 ush ct\_data\_s::freq

6.281.1.6 ush ct\_data\_s::len

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

## 6.282 activemq::commands::DataArrayResponse Class Reference

#include <src/main/activemq/commands/DataArrayResponse.h> Inheritance diagram for activemq::commands::DataArrayResponse:

### Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataArrayResponse \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &data)

### Static Public Attributes

- static const unsigned char **ID\_DATAARRAYRESPONSE** = 33

### Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**



## 6.282.1 Constructor & Destructor Documentation

**6.282.1.1** `activemq::commands::DataArrayResponse::DataArrayResponse ()`

**6.282.1.2** `virtual activemq::commands::DataArrayResponse::~~DataArrayResponse () [virtual]`

## 6.282.2 Member Function Documentation

**6.282.2.1** `virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.282.2.2** `virtual void activemq::commands::DataArrayResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.282.2.3** `virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p. 3286).

- 6.282.2.4 `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () [virtual]`
- 6.282.2.5 `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const [virtual]`
- 6.282.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Response** (p. 3287).

- 6.282.2.7 `virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & data) [virtual]`
- 6.282.2.8 `virtual std::string activemq::commands::DataArrayResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3287).

### 6.282.3 Field Documentation

- 6.282.3.1 `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]`
- 6.282.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID__ - DATAARRAYRESPONSE = 33 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

## 6.283 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1531).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.283.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1531). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.283.2 Constructor & Destructor Documentation

**6.283.2.1** `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

**6.283.2.2** `virtual activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

## 6.283.3 Member Function Documentation

**6.283.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.283.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.283.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.283.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.283.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.283.3.6** virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3303).

**6.283.3.7** virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3304).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**

## 6.284 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1535).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.284.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1535). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.284.2 Constructor & Destructor Documentation

**6.284.2.1** `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

**6.284.2.2** `virtual activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

## 6.284.3 Member Function Documentation

**6.284.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.284.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.284.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.284.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.284.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.284.3.6** virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3313).

**6.284.3.7** virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3314).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**

## 6.285 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1539).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.285.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1539). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.285.2 Constructor & Destructor Documentation

**6.285.2.1** `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayR  
( ) [inline]`

**6.285.2.2** `virtual  
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~~DataArray  
( ) [inline, virtual]`

## 6.285.3 Member Function Documentation

**6.285.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.285.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStruct  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.285.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (   
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.285.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.285.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.285.3.6** virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3298).

**6.285.3.7** virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**

## 6.286 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1543).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.286.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1543). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.286.2 Constructor & Destructor Documentation

**6.286.2.1** `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

**6.286.2.2** `virtual activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

## 6.286.3 Member Function Documentation

**6.286.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.286.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.286.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.286.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).

**6.286.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).

**6.286.3.6** virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p.3318).

**6.286.3.7** virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p.3319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataArrayResponseMarshaller.h**

## 6.287 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1547).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.287.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1547). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.287.2 Constructor & Destructor Documentation

**6.287.2.1** `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

**6.287.2.2** `virtual activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

## 6.287.3 Member Function Documentation

**6.287.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.287.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.287.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.287.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.287.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.287.3.6** virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3308).

**6.287.3.7** virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3309).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataArrayResponseMarshaller.h**

## 6.288 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1551).

#include <src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.288.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **DataArrayResponseMarshaller** (p.1551). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.288.2 Constructor & Destructor Documentation

**6.288.2.1** `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

**6.288.2.2** `virtual activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

## 6.288.3 Member Function Documentation

**6.288.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.288.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.288.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.288.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.288.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.288.3.6** virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3323).

**6.288.3.7** virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3324).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataArrayResponseMarshaller.h**

## 6.289 decaf::util::zip::DataFormatException Class Reference

#include <src/main/decaf/util/zip/DataFormatException.h> Inheritance diagram for decaf::util::zip::DataFormatException:

### Public Member Functions

- **DataFormatException** () throw ()  
*Default Constructor.*
- **DataFormatException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **DataFormatException** (const DataFormatException &ex) throw ()  
*Copy Constructor.*
- **DataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **DataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **DataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **DataFormatException** \* clone () const  
*Clones this exception.*
- virtual ~**DataFormatException** () throw ()

### 6.289.1 Constructor & Destructor Documentation

#### 6.289.1.1 decaf::util::zip::DataFormatException::DataFormatException () throw () [inline]

Default Constructor.

#### 6.289.1.2 decaf::util::zip::DataFormatException::DataFormatException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

### 6.289.1.3 decaf::util::zip::DataFormatException::DataFormatException (const DataFormatException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy, which is an instance of this type

### 6.289.1.4 decaf::util::zip::DataFormatException::DataFormatException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.289.1.5 decaf::util::zip::DataFormatException::DataFormatException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.289.1.6 decaf::util::zip::DataFormatException::DataFormatException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor.

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.289.1.7** virtual decaf::util::zip::DataFormatException::~DataFormatException ()  
throw () [inline, virtual]

## 6.289.2 Member Function Documentation

**6.289.2.1** virtual DataFormatException\* decaf::util::zip::DataFormatException::clone () const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**DataFormatException.h**

## 6.290 decaf::io::DataInput Class Reference

The **DataInput** (p.1558) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

```
#include <src/main/decaf/io/DataInput.h>
```

### Public Member Functions

- virtual **~DataInput** ()
- virtual bool **readBoolean** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*
- virtual double **readDouble** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*
- virtual int **readInt** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns an int value.*
- virtual long long **readLong** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a long value.*
- virtual short **readShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns a short value.*
- virtual unsigned short **readUnsignedShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns an int value in the range 0 through 65535.*
- virtual std::string **readString** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*
- virtual std::string **readLine** ()=0 throw ( decaf::io::IOException )  
*Reads the next line of text from the input stream.*

- virtual std::string **readUTF** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )

*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*

- virtual void **readFully** (unsigned char \*buffer, int size)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads some bytes from an input stream and stores them into the buffer array buffer.*

- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads length bytes from an input stream.*

- virtual long long **skipBytes** (long long num)=0 throw ( io::IOException )

*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.290.1 Detailed Description

The **DataInput** (p. 1558) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types. There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 1825) is thrown. If any byte cannot be read for any reason other than end of file, an **IOException** (p. 2142) other than **EOFException** (p. 1825) is thrown. for example, an **IOException** (p. 2142) may be thrown if the underlying input stream has been closed.

See also:

**DataOutput** (p. 1574)

**DataInputStream** (p. 1566)

Since:

1.0

### 6.290.2 Constructor & Destructor Documentation

- 6.290.2.1 virtual decaf::io::DataInput::~~DataInput () [inline, virtual]

### 6.290.3 Member Function Documentation

- 6.290.3.1 virtual bool decaf::io::DataInput::readBoolean () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

**Returns:**

the boolean value of the read in byte (0=false, 1=true).

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.2 virtual char decaf::io::DataInput::readByte () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads and returns one input byte. The byte is treated as a signed value in the range -128 through 127, inclusive.

**Returns:**

the 8-bit value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.3 virtual char decaf::io::DataInput::readChar () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads an input char and returns the char value. A ascii char is made up of one bytes. This returns the same result as `readByte`

**Returns:**

the 8 bit char read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.4 virtual double decaf::io::DataInput::readDouble () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads eight input bytes and returns a double value. It does this by first constructing a long long value in exactly the manner of the `readLong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

**Returns:**

the double value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.



### 6.290.3.5 virtual float decaf::io::DataInput::readFloat () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]

Reads four input bytes and returns a float value. It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

#### Returns:

the float value read.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

### 6.290.3.6 virtual void decaf::io::DataInput::readFully (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Reads length bytes from an input stream. This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1825) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2142) other than **EOFException** (p. 1825) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[off], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

#### Parameters:

*buffer* The byte array to insert read data into.

*size* The size in bytes of the given byte buffer.

*offset* The location in buffer to start writing.

*length* The number of bytes to read from the buffer.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

*NullPointerException* if the buffer is NULL.

*IndexOutOfBoundsException* if the offset + length > size, or an int param is negative.

### 6.290.3.7 virtual void decaf::io::DataInput::readFully (unsigned char \* *buffer*, int *size*) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Reads some bytes from an input stream and stores them into the buffer array buffer. The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: \* buffer's size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1825) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2142) other than **EOFException** (p. 1825) is thrown.

If buffer size is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

**Parameters:**

*buffer* The byte array to insert read data into.

*size* The size in bytes of the given byte buffer.

**Exceptions:**

**IOException** (p. 2142) if an I/O Error occurs.

**EOFException** (p. 1825) if the end of input is reached.

**IndexOutOfBoundsException** if the size value is negative.

**6.290.3.8 virtual int decaf::io::DataInput::readInt () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads four input bytes and returns an int value. Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$((a \& 0xff) \ll 24) \mid ((b \& 0xff) \ll 16) \mid ((c \& 0xff) \ll 8) \mid (d \& 0xff)$

**Returns:**

the int value read.

**Exceptions:**

**IOException** (p. 2142) if an I/O Error occurs.

**EOFException** (p. 1825) if the end of input is reached.

**6.290.3.9 virtual std::string decaf::io::DataInput::readLine () throw ( decaf::io::IOException ) [pure virtual]**

Reads the next line of text from the input stream. It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' is

encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character ' is

encountered, then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

**Returns:**

the next line of text read from the input stream or empty string if at EOF.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

**6.290.3.10 virtual long long decaf::io::DataInput::readLong () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads eight input bytes and returns a long value. Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$$(((\text{long})(a \ \& \ 0\text{xff}) \ll 56) \mid ((\text{long})(b \ \& \ 0\text{xff}) \ll 48) \mid ((\text{long})(c \ \& \ 0\text{xff}) \ll 40) \mid ((\text{long})(d \ \& \ 0\text{xff}) \ll 32) \mid ((\text{long})(e \ \& \ 0\text{xff}) \ll 24) \mid ((\text{long})(f \ \& \ 0\text{xff}) \ll 16) \mid ((\text{long})(g \ \& \ 0\text{xff}) \ll 8) \mid ((\text{long})(h \ \& \ 0\text{xff})))$$
**Returns:**

the 64 bit long long read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.11 virtual short decaf::io::DataInput::readShort () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads two input bytes and returns a short value. Let a be the first byte read and b be the second byte. The value returned is:

$$(\text{short})((a \ll 8) \mid (b \ \& \ 0\text{xff}))$$
**Returns:**

the 16 bit short value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.12 virtual std::string decaf::io::DataInput::readString () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]**

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

**Returns:**

string object containing the string read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.13** `virtual unsigned char decaf::io::DataInput::readUnsignedByte () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

**Returns:**

the 8 bit unsigned value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.14** `virtual unsigned short decaf::io::DataInput::readUnsignedShort () throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads two input bytes and returns an int value in the range 0 through 65535. Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) \mid (b \& 0xff)$

**Returns:**

the 16 bit unsigned short read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.290.3.15** `virtual std::string decaf::io::DataInput::readUTF () throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException ) [pure virtual]`

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException. This method reads String value written from a Java **DataOutputStream** (p. 1579) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

**Returns:**

The decoded string read from stream.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

***EOFException*** (p. 1825) if the end of input is reached.

***UTFDataFormatException*** (p. 3966) if the bytes are not valid modified UTF-8 values.

**6.290.3.16 virtual long long decaf::io::DataInput::skipBytes (long long *num*) throw (io::IOException) [pure virtual]**

Makes an attempt to skip over *n* bytes of data from the input stream, discarding the skipped bytes. However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1825). The actual number of bytes skipped is returned.

**Parameters:**

***num*** The number of bytes to skip over.

**Returns:**

the total number of bytes skipped.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInput.h**

## 6.291 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

#include <src/main/decaf/io/DataInputStream.h> Inheritance diagram for decaf::io::DataInputStream:

### Public Member Functions

- **DataInputStream** (**InputStream** \*inputStream, bool own=false)  
*Creates a **DataInputStream** (p. 1566) that uses the specified underlying **InputStream** (p. 2043).*
- virtual ~**DataInputStream** ()
- virtual bool **readBoolean** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*
- virtual double **readDouble** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*
- virtual int **readInt** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns an int value.*
- virtual long long **readLong** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a long value.*
- virtual short **readShort** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns a short value.*
- virtual unsigned short **readUnsignedShort** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns an int value in the range 0 through 65535.*

- virtual std::string **readString** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*
- virtual std::string **readLine** () throw ( decaf::io::IOException )  
*Reads the next line of text from the input stream.*
- virtual std::string **readUTF** () throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )  
*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*
- virtual void **readFully** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads some bytes from an input stream and stores them into the buffer array buffer.*
- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads length bytes from an input stream.*
- virtual long long **skipBytes** (long long num) throw ( io::IOException )  
*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.291.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2043) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1566) os = new DataInputStream (p. 1566)( new InputStream()
(p. 2045), true )
```

Since:

1.0

### 6.291.2 Constructor & Destructor Documentation

#### 6.291.2.1 decaf::io::DataInputStream::DataInputStream (InputStream \* inputStream, bool own = false)

Creates a **DataInputStream** (p. 1566) that uses the specified underlying **InputStream** (p. 2043).

Parameters:

*inputStream* the **InputStream** (p. 2043) instance to wrap.

*own* indicates if this class owns the wrapped string defaults to false.

**6.291.2.2** virtual `decaf::io::DataInputStream::~~DataInputStream ()` [virtual]

### 6.291.3 Member Function Documentation

**6.291.3.1** virtual `bool decaf::io::DataInputStream::readBoolean () throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

#### Returns:

the boolean value of the read in byte (0=false, 1=true).

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.291.3.2** virtual `char decaf::io::DataInputStream::readByte () throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads and returns one input byte. The byte is treated as a signed value in the range -128 through 127, inclusive.

#### Returns:

the 8-bit value read.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.291.3.3** virtual `char decaf::io::DataInputStream::readChar () throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads an input char and returns the char value. A ascii char is made up of one bytes. This returns the same result as `readByte`

#### Returns:

the 8 bit char read.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.



#### 6.291.3.4 virtual double decaf::io::DataInputStream::readDouble () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads eight input bytes and returns a double value. It does this by first constructing a long long value in exactly the manner of the readlong method, then converting this long value to a double in exactly the manner of the method Double::longBitsToDouble.

##### Returns:

the double value read.

##### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

#### 6.291.3.5 virtual float decaf::io::DataInputStream::readFloat () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads four input bytes and returns a float value. It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

##### Returns:

the float value read.

##### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

#### 6.291.3.6 virtual void decaf::io::DataInputStream::readFully (unsigned char \* buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads length bytes from an input stream. This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1825) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2142) other than **EOFException** (p. 1825) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

##### Parameters:

*buffer* The byte array to insert read data into.

*size* The size in bytes of the given byte buffer.

*offset* The location in buffer to start writing.

**length** The number of bytes to read from the buffer.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

***EOFException*** (p. 1825) if the end of input is reached.

***NullPointerException*** if the buffer is NULL.

***IndexOutOfBoundsException*** if the offset + length > size.

**6.291.3.7** **virtual void decaf::io::DataInputStream::readFully (unsigned char \*  
buffer, int size) throw ( decaf::io::IOException, decaf::io::EOFException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]**

Reads some bytes from an input stream and stores them into the buffer array buffer. The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: \* buffer's size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1825) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2142) other than **EOFException** (p. 1825) is thrown.

If buffer size is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

**Parameters:**

**buffer** The byte array to insert read data into.

**size** The size in bytes of the given byte buffer.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

***EOFException*** (p. 1825) if the end of input is reached.

***IndexOutOfBoundsException*** if the size value is negative.

**6.291.3.8** **virtual int decaf::io::DataInputStream::readInt () throw (  
decaf::io::IOException, decaf::io::EOFException ) [virtual]**

Reads four input bytes and returns an int value. Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff)$

**Returns:**

the int value read.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

***EOFException*** (p. 1825) if the end of input is reached.

### 6.291.3.9 virtual std::string decaf::io::DataInputStream::readLine () throw ( decaf::io::IOException ) [virtual]

Reads the next line of text from the input stream. It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' ' is encountered, it is discarded and reading ceases. If the character " " is encountered, it is discarded and, if the following byte converts to the character ' ' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

' is encountered, it is discarded and reading ceases. If the character " " is encountered, it is discarded and, if the following byte converts to the character ' ' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

' and " " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

#### Returns:

the next line of text read from the input stream or empty string if at EOF.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

### 6.291.3.10 virtual long long decaf::io::DataInputStream::readLong () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads eight input bytes and returns a long value. Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$$(((\text{long})(a \& 0\text{xff}) << 56) | ((\text{long})(b \& 0\text{xff}) << 48) | ((\text{long})(c \& 0\text{xff}) << 40) | ((\text{long})(d \& 0\text{xff}) << 32) | ((\text{long})(e \& 0\text{xff}) << 24) | ((\text{long})(f \& 0\text{xff}) << 16) | ((\text{long})(g \& 0\text{xff}) << 8) | ((\text{long})(h \& 0\text{xff})))$$

#### Returns:

the 64 bit long long read.

#### Exceptions:

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

### 6.291.3.11 virtual short decaf::io::DataInputStream::readShort () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads two input bytes and returns a short value. Let a be the first byte read and b be the second byte. The value returned is:

$$(\text{short})((a << 8) | (b \& 0\text{xff}))$$

**Returns:**

the 16 bit short value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

### 6.291.3.12 virtual std::string decaf::io::DataInputStream::readString () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

**Returns:**

string object containing the string read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

### 6.291.3.13 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

**Returns:**

the 8 bit unsigned value read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

### 6.291.3.14 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535. Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) | (b \& 0xff)$

**Returns:**

the 16 bit unsigned short read.

**Exceptions:**

*IOException* (p. 2142) if an I/O Error occurs.

*EOFException* (p. 1825) if the end of input is reached.

**6.291.3.15**    `virtual std::string decaf::io::DataInputStream::readUTF ()`  
                 `throw ( decaf::io::IOException, decaf::io::EOFException,`  
                 `decaf::io::UTFDataFormatException ) [virtual]`

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a `UTFFormatException`. This method reads String value written from a Java **DataOutputStream** (p. 1579) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

**Returns:**

The decoded string read from stream.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

***EOFException*** (p. 1825) if the end of input is reached.

***UTFDataFormatException*** (p. 3966) if the bytes are not valid modified UTF-8 values.

**6.291.3.16**    `virtual long long decaf::io::DataInputStream::skipBytes (long long num)`  
                 `throw ( io::IOException ) [virtual]`

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes. However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1825). The actual number of bytes skipped is returned.

**Parameters:**

***num*** The number of bytes to skip over.

**Returns:**

the total number of bytes skipped.

**Exceptions:**

***IOException*** (p. 2142) if an I/O Error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInputStream.h`

## 6.292 decaf::io::DataOutput Class Reference

The **DataOutput** (p. 1574) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

```
#include <src/main/decaf/io/DataOutput.h>
```

### Public Member Functions

- virtual **~DataOutput** ()
- virtual void **writeBoolean** (bool value)=0 throw ( decaf::io::IOException )  
*Writes a boolean to the underlying output stream as a 1-byte value.*
- virtual void **writeByte** (unsigned char value)=0 throw ( decaf::io::IOException )  
*Writes out a byte to the underlying output stream as a 1-byte value.*
- virtual void **writeShort** (short value)=0 throw ( decaf::io::IOException )  
*Writes a short to the underlying output stream as two bytes, high byte first.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( decaf::io::IOException )  
*Writes a unsigned short to the bytes message stream as a 2 byte value.*
- virtual void **writeChar** (char value)=0 throw ( decaf::io::IOException )  
*Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.*
- virtual void **writeInt** (int value)=0 throw ( decaf::io::IOException )  
*Writes an int to the underlying output stream as four bytes, high byte first.*
- virtual void **writeLong** (long long value)=0 throw ( decaf::io::IOException )  
*Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.*
- virtual void **writeFloat** (float value)=0 throw ( decaf::io::IOException )  
*Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.*
- virtual void **writeDouble** (double value)=0 throw ( decaf::io::IOException )  
*Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.*
- virtual void **writeBytes** (const std::string &value)=0 throw ( decaf::io::IOException )  
*Writes out the string to the underlying output stream as a sequence of bytes.*
- virtual void **writeChars** (const std::string &value)=0 throw ( decaf::io::IOException )  
*Writes a string to the underlying output stream as a sequence of characters.*
- virtual void **writeUTF** (const std::string &value)=0 throw ( decaf::io::IOException, decaf::io::UTFDataFormatException )  
*Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.*

## 6.292.1 Detailed Description

The **DataOutput** (p. 1574) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream. There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 2142).

See also:

**DataInput** (p. 1558)

**DataOutputStream** (p. 1579)

Since:

1.0

## 6.292.2 Constructor & Destructor Documentation

**6.292.2.1** virtual decaf::io::DataOutput::~~DataOutput () [inline, virtual]

## 6.292.3 Member Function Documentation

**6.292.3.1** virtual void decaf::io::DataOutput::writeBoolean (bool *value*) throw ( decaf::io::IOException ) [pure virtual]

Writes a boolean to the underlying output stream as a 1-byte value. The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

**Parameters:**

*value* The boolean to write as a byte (1=true, 0=false).

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.2** virtual void decaf::io::DataOutput::writeByte (unsigned char *value*) throw ( decaf::io::IOException ) [pure virtual]

Writes out a byte to the underlying output stream as a 1-byte value. If no exception is thrown, the counter written is incremented by 1.

**Parameters:**

*value* The unsigned char value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.3 virtual void decaf::io::DataOutput::writeBytes (const std::string & *value*) throw ( decaf::io::IOException ) [pure virtual]**

Writes out the string to the underlying output stream as a sequence of bytes. Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

**Parameters:**

*value* The vector of bytes to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.4 virtual void decaf::io::DataOutput::writeChar (char *value*) throw ( decaf::io::IOException ) [pure virtual]**

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

**Parameters:**

*value* The signed char value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.5 virtual void decaf::io::DataOutput::writeChars (const std::string & *value*) throw ( decaf::io::IOException ) [pure virtual]**

Writes a string to the underlying output stream as a sequence of characters. Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

**Parameters:**

*value* The string value to write as raw bytes.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.6 virtual void decaf::io::DataOutput::writeDouble (double *value*) throw ( decaf::io::IOException ) [pure virtual]**

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 8.



**Parameters:**

*value* The 64bit double value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.7 virtual void decaf::io::DataOutput::writeFloat (float *value*) throw ( decaf::io::IOException ) [pure virtual]**

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 4.

**Parameters:**

*value* The 32bit floating point value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.8 virtual void decaf::io::DataOutput::writeInt (int *value*) throw ( decaf::io::IOException ) [pure virtual]**

Writes an int to the underlying output stream as four bytes, high byte first. If no exception is thrown, the counter written is incremented by 4.

**Parameters:**

*value* The signed integer value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.9 virtual void decaf::io::DataOutput::writeLong (long long *value*) throw ( decaf::io::IOException ) [pure virtual]**

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first. If no exception is thrown, the counter written is incremented by 8.

**Parameters:**

*value* The signed 64bit long value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.10** `virtual void decaf::io::DataOutput::writeShort (short value) throw ( decaf::io::IOException )` [pure virtual]

Writes a short to the underlying output stream as two bytes, high byte first. If no exception is thrown, the counter written is incremented by 2.

**Parameters:**

*value* The signed short value to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.11** `virtual void decaf::io::DataOutput::writeUnsignedShort (unsigned short value) throw ( decaf::io::IOException )` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

**Parameters:**

*value* The unsigned short to write to the stream.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

**6.292.3.12** `virtual void decaf::io::DataOutput::writeUTF (const std::string & value) throw ( decaf::io::IOException, decaf::io::UTFDataFormatException )` [pure virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes. The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

**Parameters:**

*value* The string value value to write as modified UTF-8.

**Exceptions:**

*IOException* (p. 2142) if an I/O error is encountered.

*UTFDataFormatException* (p. 3966) if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutput.h`

## 6.293 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

#include <src/main/decaf/io/DataOutputStream.h> Inheritance diagram for decaf::io::DataOutputStream:

### Public Member Functions

- **DataOutputStream** (**OutputStream** \***outputStream**, bool **own**=false)  
*Creates a new data output stream to write data to the specified underlying output stream.*
- virtual ~**DataOutputStream** ()
- virtual long long **size** () const  
*Returns the current value of the counter written, the number of bytes written to this data output stream so far.*
- virtual void **writeBoolean** (bool value) throw ( **IOException** )
- virtual void **writeByte** (unsigned char value) throw ( **IOException** )
- virtual void **writeShort** (short value) throw ( **IOException** )
- virtual void **writeUnsignedShort** (unsigned short value) throw ( **IOException** )
- virtual void **writeChar** (char value) throw ( **IOException** )
- virtual void **writeInt** (int value) throw ( **IOException** )
- virtual void **writeLong** (long long value) throw ( **IOException** )
- virtual void **writeFloat** (float value) throw ( **IOException** )
- virtual void **writeDouble** (double value) throw ( **IOException** )
- virtual void **writeBytes** (const std::string &value) throw ( **IOException** )

- virtual void **writeChars** (const std::string &value) throw ( IOException )
- virtual void **writeUTF** (const std::string &value) throw ( IOException, UTFDataFormatException )

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \***buffer**, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

## Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

### 6.293.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

### 6.293.2 Constructor & Destructor Documentation

#### 6.293.2.1 decaf::io::DataOutputStream::DataOutputStream (OutputStream \* *outputStream*, bool *own* = false)

Creates a new data output stream to write data to the specified underlying output stream.

#### Parameters:

*outputStream* a stream to wrap with this one.

*own* true if this objects owns the stream that it wraps.

#### 6.293.2.2 virtual decaf::io::DataOutputStream::~~DataOutputStream () [virtual]

### 6.293.3 Member Function Documentation

#### 6.293.3.1 virtual void decaf::io::DataOutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.293.3.2** virtual void decaf::io::DataOutputStream::doWriteByte (unsigned char *value*) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.293.3.3** virtual long long decaf::io::DataOutputStream::size () const [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far. If the counter overflows, it will be wrapped to **decaf::lang::Long::MAX\_VALUE** (p. 2433).

**Returns:**

the value of the written field.

- 6.293.3.4 virtual void decaf::io::DataOutputStream::writeBoolean (bool *value*)  
throw ( IOException ) [virtual]
- 6.293.3.5 virtual void decaf::io::DataOutputStream::writeByte (unsigned char  
*value*) throw ( IOException ) [virtual]
- 6.293.3.6 virtual void decaf::io::DataOutputStream::writeBytes (const std::string &  
*value*) throw ( IOException ) [virtual]
- 6.293.3.7 virtual void decaf::io::DataOutputStream::writeChar (char *value*) throw (   
IOException ) [virtual]
- 6.293.3.8 virtual void decaf::io::DataOutputStream::writeChars (const std::string  
& *value*) throw ( IOException ) [virtual]
- 6.293.3.9 virtual void decaf::io::DataOutputStream::writeDouble (double *value*)  
throw ( IOException ) [virtual]
- 6.293.3.10 virtual void decaf::io::DataOutputStream::writeFloat (float *value*) throw  
( IOException ) [virtual]
- 6.293.3.11 virtual void decaf::io::DataOutputStream::writeInt (int *value*) throw (   
IOException ) [virtual]
- 6.293.3.12 virtual void decaf::io::DataOutputStream::writeLong (long long *value*)  
throw ( IOException ) [virtual]
- 6.293.3.13 virtual void decaf::io::DataOutputStream::writeShort (short *value*)  
throw ( IOException ) [virtual]
- 6.293.3.14 virtual void decaf::io::DataOutputStream::writeUnsignedShort  
(unsigned short *value*) throw ( IOException ) [virtual]
- 6.293.3.15 virtual void decaf::io::DataOutputStream::writeUTF (const std::string  
& *value*) throw ( IOException, UTFDataFormatException ) [virtual]

## 6.293.4 Field Documentation

- 6.293.4.1 unsigned char decaf::io::DataOutputStream::buffer[8] [protected]
- 6.293.4.2 long long decaf::io::DataOutputStream::written [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutputStream.h**

## 6.294 activemq::commands::DataResponse Class Reference

#include <src/main/activemq/commands/DataResponse.h> Inheritance diagram for activemq::commands::DataResponse:

### Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataResponse \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

### Static Public Attributes

- static const unsigned char **ID\_DATARESPONSE** = 32

### Protected Attributes

- **Pointer**< **DataStructure** > data

## 6.294.1 Constructor & Destructor Documentation

**6.294.1.1** `activemq::commands::DataResponse::DataResponse ()`

**6.294.1.2** `virtual activemq::commands::DataResponse::~~DataResponse ()`  
[virtual]

## 6.294.2 Member Function Documentation

**6.294.2.1** `virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.3286).

**6.294.2.2** `virtual void activemq::commands::DataResponse::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::Response` (p.3286).

**6.294.2.3** `virtual bool activemq::commands::DataResponse::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p.1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.3286).



- 6.294.2.4** `virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData () [virtual]`
- 6.294.2.5** `virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData () const [virtual]`
- 6.294.2.6** `virtual unsigned char activemq::commands::DataResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Response** (p. 3287).

- 6.294.2.7** `virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & data) [virtual]`
- 6.294.2.8** `virtual std::string activemq::commands::DataResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3287).

### 6.294.3 Field Documentation

- 6.294.3.1** `Pointer<DataStructure> activemq::commands::DataResponse::data [protected]`
- 6.294.3.2** `const unsigned char activemq::commands::DataResponse::ID_ - DATARESPONSE = 32 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataResponse.h`

## 6.295 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1586).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.295.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1586).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.295.2 Constructor & Destructor Documentation

**6.295.2.1** `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMarshaller()` [inline]

**6.295.2.2** `virtual activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

## 6.295.3 Member Function Documentation

**6.295.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.295.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.295.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.295.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.295.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.295.3.6** virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3308).

**6.295.3.7** virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3309).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataResponseMarshaller.h**

## 6.296 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1590).

#include <src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.296.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1590).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.296.2 Constructor & Destructor Documentation

**6.296.2.1** `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::DataResponseMarshaller()` [inline]

**6.296.2.2** `virtual activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

## 6.296.3 Member Function Documentation

**6.296.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.296.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.296.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.296.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.296.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).



**6.296.3.6** virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3323).

**6.296.3.7** virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3324).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataResponseMarshaller.h**

## 6.297 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1594).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.297.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1594).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.297.2 Constructor & Destructor Documentation

**6.297.2.1** `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMarshaller()` [inline]

**6.297.2.2** `virtual activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

## 6.297.3 Member Function Documentation

**6.297.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.297.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.297.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.297.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.297.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.297.3.6** virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3303).

**6.297.3.7** virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3304).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**

## 6.298 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1598).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.298.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1598).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.298.2 Constructor & Destructor Documentation

**6.298.2.1** `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMar  
( ) [inline]`

**6.298.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseM  
( ) [inline, virtual]`

## 6.298.3 Member Function Documentation

**6.298.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.298.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.298.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.298.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.298.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).



**6.298.3.6** virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3313).

**6.298.3.7** virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3314).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**

## 6.299 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1602).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.299.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1602).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.299.2 Constructor & Destructor Documentation

**6.299.2.1** `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMarshaller()` [inline]

**6.299.2.2** `virtual activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

## 6.299.3 Member Function Documentation

**6.299.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.299.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.299.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.299.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.299.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.299.3.6** virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3298).

**6.299.3.7** virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**

## 6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1606).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.300.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DataResponseMarshaller** (p.1606).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.300.2 Constructor & Destructor Documentation

**6.300.2.1** `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMar  
( ) [inline]`

**6.300.2.2** `virtual  
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseM  
( ) [inline, virtual]`

## 6.300.3 Member Function Documentation

**6.300.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`  
(p. 3316).

**6.300.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`  
(p. 3316).

**6.300.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.300.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).

**6.300.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).



**6.300.3.6** virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3318).

**6.300.3.7** virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataResponseMarshaller.h**

## 6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that **marshal** (p.107) **commands** (p. 87) for Openwire.

#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::DataStreamMarshaller:

### Public Member Functions

- virtual **~DataStreamMarshaller** ()
- virtual unsigned char **getDataStructureType** () const =0  
*Gets the DataStructureType that this class marshals/unmarshals.*
- virtual **commands::DataStructure \* createObject** () const =0  
*Creates a new instance of the class that this class is a marshaling director for.*
- virtual int **tightMarshal1** (**OpenWireFormat \*format**, **commands::DataStructure \*command**, **utils::BooleanStream \*bs**)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*format**, **commands::DataStructure \*command**, **decaf::io::DataOutputStream \*ds**, **utils::BooleanStream \*bs**)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*format**, **commands::DataStructure \*command**, **decaf::io::DataInputStream \*dis**, **utils::BooleanStream \*bs**)=0 throw ( **decaf::io::IOException** )  
*Tight Un-marhsal to the given stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*format**, **commands::DataStructure \*command**, **decaf::io::DataOutputStream \*ds**)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*format**, **commands::DataStructure \*command**, **decaf::io::DataInputStream \*dis**)=0 throw ( **decaf::io::IOException** )  
*Loose Un-marhsal to the given stream.*

### 6.301.1 Detailed Description

Base class for all classes that **marshal** (p.107) **commands** (p. 87) for Openwire.

## 6.301.2 Constructor & Destructor Documentation

**6.301.2.1** virtual  
activemq::wireformat::openwire::marshal::DataStreamMarshaller::~~DataStreamMarshaller  
() [inline, virtual]

## 6.301.3 Member Function Documentation

**6.301.3.1** virtual commands::DataStructure\* ac-  
tivemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject  
() const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

### Returns:

newly allocated Command

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller  
(p. 214), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller  
(p. 254), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller  
(p. 379), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller  
(p. 406), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller  
(p. 451), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller  
(p. 496), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller  
(p. 557), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller  
(p. 614), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller  
(p. 647), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller  
(p. 676), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller  
(p. 704), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 881),  
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 913), ac-  
tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1281),  
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1313),  
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1344),  
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1374),  
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1419),  
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1448),  
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1481), ac-  
tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1510), ac-  
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1544),  
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1607),  
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1745),  
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1779), ac-  
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1863),  
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1957),  
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2111),  
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2180),  
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2209),  
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2232), ac-  
tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2263),  
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2290),  
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller  
(p. 2325), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller  
(p. 2372), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller

(p. 2582), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`  
 (p. 2623), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
 (p. 2652), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2689),  
`activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2761), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2817),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2943),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3061),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3093),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3110),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3211), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
 (p. 3228), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
 (p. 3259), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3404), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3420), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3482), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3681),  
`activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3856),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 4009),  
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4049),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`  
 (p. 222), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
 (p. 270), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
 (p. 391), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
 (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
 (p. 463), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`  
 (p. 508), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
 (p. 569), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`  
 (p. 626), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`  
 (p. 655), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
 (p. 688), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`  
 (p. 716), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 893),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 925), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1293),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1301),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1332),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1362),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1407),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1436),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1469), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1498), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1532),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1595),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1733),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1767), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1847),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1945),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2099),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2164),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2193),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2216), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2247),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2274),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`

(p. 2313), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2356), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2570), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2607), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2640), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2669), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2745), `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2797), `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2927), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3041), `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3073), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3106), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3199), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3236), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3263), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301), `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3384), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3428), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3478), `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3697), `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3872), `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4001), `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4041), `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 210), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 250), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 553), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 610), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 639), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 668), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 696), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 873), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 905), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1305), `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1336), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1366), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1473), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1536), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1737), `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1771), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1851), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1949), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2103), `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2172), `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2197), `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2220), `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2251),

activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2278),  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
 (p. 2309), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
 (p. 2360), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
 (p. 2574), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
 (p. 2611), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
 (p. 2644), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2681),  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2753), ac-  
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2809),  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2935),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3049),  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3081),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3118),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3207), ac-  
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
 (p. 3232), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
 (p. 3267), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3311),  
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3400), ac-  
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3424), ac-  
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3490), ac-  
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3677),  
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3860),  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4013),  
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4053),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
 (p. 218), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
 (p. 258), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
 (p. 383), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
 (p. 410), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
 (p. 455), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
 (p. 500), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
 (p. 561), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
 (p. 618), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
 (p. 643), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
 (p. 672), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
 (p. 700), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 877),  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 909), ac-  
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1277),  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1309),  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1340),  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1370),  
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1415),  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1444),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1477), ac-  
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1506), ac-  
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1540),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1603),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1741),  
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1775), ac-  
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1859),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1953),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2107),  
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2176),  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2205),

activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2228),  
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2259),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2282),  
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller  
 (p. 2321), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller  
 (p. 2368), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller  
 (p. 2578), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller  
 (p. 2619), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
 (p. 2648), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2673),  
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2757),  
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2813),  
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2939),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3045),  
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3077),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3102),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3219),  
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3248), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3255), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3296),  
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3388),  
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3432),  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3494),  
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3689),  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3868),  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4005),  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4045),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
 (p. 226), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
 (p. 262), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
 (p. 387), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
 (p. 414), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
 (p. 459), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
 (p. 504), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
 (p. 565), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
 (p. 622), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
 (p. 651), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
 (p. 680), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
 (p. 708), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 885),  
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 917),  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1285),  
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1317),  
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1348),  
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1378),  
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1423),  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1452),  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1485),  
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1514),  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1548),  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1587),  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1753),  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1783),  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1855),  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1961),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2115),

activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2168),  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2189),  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2236),  
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2255),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2286),  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
 (p. 2317), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller  
 (p. 2364), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
 (p. 2586), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
 (p. 2615), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
 (p. 2656), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2677),  
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2749),  
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2805),  
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2931),  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3053),  
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3085),  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3114),  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3215),  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
 (p. 3244), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
 (p. 3275), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3306),  
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3396),  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3416),  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3486),  
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3685),  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3852),  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3993),  
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4057),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
 (p. 230), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
 (p. 266), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
 (p. 395), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
 (p. 422), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
 (p. 467), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
 (p. 512), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
 (p. 573), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
 (p. 630), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
 (p. 659), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
 (p. 684), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
 (p. 712), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 889),  
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 921),  
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1289),  
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1321),  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1352),  
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1382),  
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1427),  
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1456),  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1489),  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1518),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1552),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1591),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1749),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1763),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1843),



activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1941),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2095),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2160),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2201),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2224),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2243),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2270),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2305), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2352), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2566), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2627), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2636), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2685),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2765),  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2801),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2923),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3057),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3089),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3122),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3203),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3240), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3271), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3321),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3392),  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3412),  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3474),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3693),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3864),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3997), and  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4037).

### 6.301.3.2 virtual unsigned char ac-

tivemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType  
 () const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

#### Returns:

byte Id of this classes DataStructureType

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller  
 (p. 214), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller  
 (p. 254), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller  
 (p. 379), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller  
 (p. 406), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller  
 (p. 451), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller  
 (p. 496), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller  
 (p. 557), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller  
 (p. 614), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller  
 (p. 647), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller  
 (p. 676), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller  
 (p. 704), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 881),

activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 913),  
 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1281),  
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1313),  
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1344),  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1374),  
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1419),  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1448),  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1481),  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1510),  
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1544),  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1607),  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1745),  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1779),  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1863),  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1957),  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2111),  
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2180),  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2209),  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2232),  
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2263),  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2290),  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2325),  
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2372),  
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2582),  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2623),  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2652),  
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2689),  
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2761),  
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2817),  
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2943),  
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3061),  
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3093),  
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3110),  
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3211),  
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3228),  
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3259),  
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3316),  
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3404),  
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3420),  
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3482),  
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3681),  
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3856),  
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4009),  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4049),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 222),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 270),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 391),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 418),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 463),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 508),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 569),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 626),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 655),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller

(p. 688),       activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
 (p. 716),   activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller   (p. 893),  
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller   (p. 925),   ac-  
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller   (p. 1293),  
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller   (p. 1301),  
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller   (p. 1332),  
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller   (p. 1362),  
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller   (p. 1407),  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller   (p. 1436),  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller   (p. 1469),   ac-  
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller   (p. 1498),   ac-  
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller   (p. 1532),  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller   (p. 1595),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller   (p. 1733),  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller   (p. 1767),   ac-  
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller   (p. 1847),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller   (p. 1945),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller   (p. 2099),  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller   (p. 2164),  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller   (p. 2193),  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller   (p. 2216),   ac-  
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller   (p. 2247),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller   (p. 2274),  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
 (p. 2313),   activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
 (p. 2356),       activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
 (p. 2570),   activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
 (p. 2607),   activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2640),   activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller   (p. 2669),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller   (p. 2745),   ac-  
 tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller   (p. 2797),  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller   (p. 2927),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller   (p. 3041),  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller   (p. 3073),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller   (p. 3106),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller   (p. 3199),   ac-  
 tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3236),   activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3263),   activemq::wireformat::openwire::marshal::v2::ResponseMarshaller   (p. 3301),  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller   (p. 3384),   ac-  
 tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller   (p. 3428),   ac-  
 tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller   (p. 3478),   ac-  
 tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller   (p. 3697),  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller   (p. 3872),  
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller   (p. 4001),  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller   (p. 4041),  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 210),   activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 250),   activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 375),   activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 402),   activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 447),       activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
 (p. 492),   activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 553),   activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller

(p. 610), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
 (p. 639), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
 (p. 668), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
 (p. 696), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 873),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 905), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1273),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1305),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1336),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1366),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1411),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1440),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1473), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1502), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1536),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1599),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1737),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1771), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1851),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1949),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2103),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2172),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2197),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2220), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2251),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2278),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`  
 (p. 2309), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`  
 (p. 2360), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`  
 (p. 2574), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`  
 (p. 2611), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
 (p. 2644), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2681),  
`activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2753), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2809),  
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2935),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3049),  
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3081),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3118),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3207), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
 (p. 3232), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
 (p. 3267), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311),  
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3400), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3424), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3490), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3677),  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3860),  
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4013),  
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4053),  
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
 (p. 218), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
 (p. 258), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
 (p. 383), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
 (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
 (p. 455), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`

(p. 500), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
 (p. 561), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
 (p. 618), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
 (p. 643), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
 (p. 672), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
 (p. 700), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 877),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 909), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1277),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1309),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1340),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1370),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1415),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1444),  
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1477), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1506), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1540),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1603),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1741),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1775), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1859),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1953),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2107),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2176),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2205),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2228), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2259),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2282),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
 (p. 2321), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
 (p. 2368), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
 (p. 2578), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
 (p. 2619), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
 (p. 2648), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2673),  
`activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2757), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2813),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2939),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3045),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3077),  
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3102),  
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3219), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`  
 (p. 3248), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`  
 (p. 3255), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296),  
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3388), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3432), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3494), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3689),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3868),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 4005),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4045),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
 (p. 226), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
 (p. 262), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
 (p. 387), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`

(p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
 (p. 459), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`  
 (p. 504), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
 (p. 565), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
 (p. 622), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
 (p. 651), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
 (p. 680), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`  
 (p. 708), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 885),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 917), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1285),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1317),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1348),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1378),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1423),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1452),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1485), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1514), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1548),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1587),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1753),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1783), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1855),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1961),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2115),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2168),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2189),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2236), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2255),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
 (p. 2317), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
 (p. 2364), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
 (p. 2586), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
 (p. 2615), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
 (p. 2656), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2677),  
`activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2749), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2805),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2931),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3053),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3085),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3114),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3215), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
 (p. 3244), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
 (p. 3275), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3396), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3416), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3486), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3685),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3852),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3993),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4057),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`  
 (p. 230), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`

(p. 266), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
 (p. 395), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
 (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
 (p. 467), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
 (p. 512), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
 (p. 573), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
 (p. 630), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
 (p. 659), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
 (p. 684), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
 (p. 712), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 889),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 921), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1289),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1321),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1352),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1382),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1427),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1456),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1489), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1518), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1552),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1591),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1749),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1763), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1843),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1941),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2095),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2160),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2201),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2224), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2243),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2270),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`  
 (p. 2305), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`  
 (p. 2352), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`  
 (p. 2566), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`  
 (p. 2627), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`  
 (p. 2636), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2685),  
`activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2765), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2801),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2923),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3057),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3089),  
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3122),  
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3203), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`  
 (p. 3240), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`  
 (p. 3271), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321),  
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3392), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3412), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3474), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3693),  
`activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3864), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3997), and  
`activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4037).

**6.301.3.3** virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat \* *format*, commands::DataStructure \* *command*, decaf::io::DataOutputStream \* *ds*) throw ( decaf::io::IOException )  
[pure virtual]

Tight Marshal to the given stream.

#### Parameters:

*format* - The OpenWireFormat properties  
*command* - the object to Marshal  
*ds* - DataOutputStream to **marshal** (p.107) to

#### Exceptions:

*IOException* if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 214), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 254), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 338), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 379), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 406), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 451), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 496), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 557), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 585), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 614), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 647), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 676), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 704), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 780), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 881), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 913), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1281), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1313), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1344), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1374), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1419), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1448), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1481), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1510), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1544), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1607), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1745), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1779), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1863), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1957), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2111), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2180), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2209), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2232), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2263), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2290),



activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2325),  
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2372),  
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2582),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2623),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2652),  
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2689),  
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2714),  
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2761),  
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2817),  
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2943),  
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3061),  
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3093),  
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3110),  
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3211),  
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3228),  
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3259),  
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3316),  
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3404),  
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3420),  
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3482),  
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3681),  
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3828),  
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3856),  
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4009),  
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4049),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 222),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 270),  
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 350),  
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 391),  
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 418),  
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 463),  
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 508),  
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 569),  
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 597),  
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 626),  
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 655),  
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 688),  
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 716),  
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 801),  
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 893),  
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 925),  
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1293),  
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1301),  
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1332),  
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1362),  
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1407),  
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1436),  
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1469),  
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1498),  
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1532),  
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1595),  
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1733),  
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1767),  
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1847),

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1945),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2099),  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2164),  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2193),  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2216),  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2247),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2274),  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
 (p. 2313), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
 (p. 2356), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
 (p. 2570), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
 (p. 2607), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2640), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2669),  
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2704),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2745),  
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2797),  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2927),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3041),  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3073),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3106),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3199),  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3236), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3263), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3301),  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3384),  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3428),  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3478),  
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3697),  
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3832),  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3872),  
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4001),  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4041),  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 210), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 250), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
 (p. 334), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 375), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 402), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 447), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller  
 (p. 492), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 553), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
 (p. 581), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 (p. 610), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
 (p. 639), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
 (p. 668), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
 (p. 696), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
 (p. 766), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 873),  
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 905),  
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1273),  
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1305),  
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1336),  
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1366),  
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1411),  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1440),

activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1473),  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1502),  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1536),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1599),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1737),  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1771),  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1851),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1949),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2103),  
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2172),  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2197),  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2220),  
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2251),  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2278),  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
 (p. 2309), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
 (p. 2360), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
 (p. 2574), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
 (p. 2611), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
 (p. 2644), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2681),  
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2699),  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2753),  
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2809),  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2935),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3049),  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3081),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3118),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3207),  
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
 (p. 3232), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
 (p. 3267), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3311),  
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3400),  
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3424),  
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3490),  
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3677),  
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3836),  
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3860),  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4013),  
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4053),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
 (p. 218), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
 (p. 258), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
 (p. 342), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
 (p. 383), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
 (p. 410), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
 (p. 455), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
 (p. 500), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
 (p. 561), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
 (p. 589), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
 (p. 618), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
 (p. 643), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
 (p. 672), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
 (p. 700), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
 (p. 773), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 877),

activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 909),  
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1277),  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1309),  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1340),  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1370),  
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1415),  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1444),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1477),  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1506),  
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1540),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1603),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1741),  
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1775),  
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1859),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1953),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2107),  
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2176),  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2205),  
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2228),  
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2259),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2282),  
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2321),  
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2368),  
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2578),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2619),  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2648),  
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2673),  
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2709),  
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2757),  
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2813),  
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2939),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3045),  
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3077),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3102),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3219),  
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3248),  
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3255),  
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3296),  
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3388),  
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3432),  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3494),  
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3689),  
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3840),  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3868),  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4005),  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4045),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 226),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 262),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 346),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 387),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 414),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 459),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 504),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 565), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`  
(p. 593), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
(p. 622), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
(p. 651), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
(p. 680), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`  
(p. 708), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`  
(p. 787), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 885),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 917), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1285),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1317),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1348),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1378),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1423),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1452),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1485), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1514), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1548),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1587),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1753),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1783), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1855),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1961),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2115),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2168),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2189),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2236), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2255),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
(p. 2317), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
(p. 2364), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
(p. 2586), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
(p. 2615), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
(p. 2656), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2677),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2749), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2805),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2931),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3053),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3085),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3114),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3215), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3244), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
(p. 3275), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3396), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3416), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3486), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3685),  
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3824),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3852),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3993),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4057),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`

(p. 230), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`  
 (p. 266), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`  
 (p. 354), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
 (p. 395), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
 (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
 (p. 467), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
 (p. 512), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
 (p. 573), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
 (p. 601), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
 (p. 630), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
 (p. 659), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
 (p. 684), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
 (p. 712), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
 (p. 794), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 889),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 921), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1289),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1321),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1352),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1382),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1427),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1456),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1489), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1518), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1552),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1591),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1749),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1763), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1843),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1941),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2095),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2160),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2201),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2224), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2243),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2270),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`  
 (p. 2305), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`  
 (p. 2352), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`  
 (p. 2566), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`  
 (p. 2627), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`  
 (p. 2636), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2685),  
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2719), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2765), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2801),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2923),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3057),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3089),  
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3122),  
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3203), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`  
 (p. 3240), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`  
 (p. 3271), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321),  
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3392), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3412), `ac-`

tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3474), activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3693), activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3844), activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3864), activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3997), and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4037).

**6.301.3.4** virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat \* *format*, commands::DataStructure \* *command*, decaf::io::DataInputStream \* *dis*) throw ( decaf::io::IOException ) [pure virtual]

Loose Un-marhsal to the given stream.

#### Parameters:

*format* - The OpenWireFormat properties  
*command* - the object to Un-Marshal  
*dis* - the DataInputStream to Un-Marshal from

#### Exceptions:

*IOException* if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 215), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 255), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 338), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 380), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 407), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 452), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 497), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 585), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 615), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 648), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 677), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 705), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 781), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 882), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 914), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1282), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1314), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1345), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1375), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1420), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1449), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1482), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1511), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1545), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1608), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1746), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1780), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1864),

activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1958),  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2112),  
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2181),  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2210),  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2233),  
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2264),  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2291),  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller  
 (p. 2326), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller  
 (p. 2373), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller  
 (p. 2583), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller  
 (p. 2624), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
 (p. 2653), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2690),  
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2714),  
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2762),  
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2818),  
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2944),  
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3062),  
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3094),  
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3111),  
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3212),  
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller  
 (p. 3229), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller  
 (p. 3260), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3317),  
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3405),  
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3421),  
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3483),  
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3682),  
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3828),  
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3857),  
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4010),  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4050),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller  
 (p. 223), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
 (p. 271), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
 (p. 350), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
 (p. 392), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
 (p. 419), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
 (p. 464), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
 (p. 509), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
 (p. 570), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
 (p. 597), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
 (p. 627), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
 (p. 656), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
 (p. 689), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
 (p. 717), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller  
 (p. 802), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 894),  
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 926),  
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1294),  
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1302),  
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1333),  
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1363),  
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1408),  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1437),



activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1470),  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1499),  
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1533),  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1596),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1734),  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1768),  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1848),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1946),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2100),  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2165),  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2194),  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2217),  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2248),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2275),  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
 (p. 2314), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
 (p. 2357), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
 (p. 2571), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
 (p. 2608), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2641), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2670),  
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2704),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2746),  
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2798),  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2928),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3042),  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3074),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3107),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3200),  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3237), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3264), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3302),  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3385),  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3429),  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3479),  
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3698),  
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3832),  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3873),  
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4002),  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4042),  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 211), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 251), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
 (p. 334), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 376), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 403), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 448), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
 (p. 493), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
 (p. 581), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 (p. 611), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
 (p. 640), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
 (p. 669), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
 (p. 697), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
 (p. 767), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 874),

activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 906),  
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1274),  
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1306),  
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1337),  
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1367),  
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1412),  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1441),  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1474),  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1503),  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1537),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1600),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1738),  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1772),  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1852),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1950),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2104),  
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2173),  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2198),  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2221),  
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2252),  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2279),  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 2310),  
 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 2361),  
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2575),  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2612),  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2645),  
 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2682),  
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2699),  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2754),  
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2810),  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2936),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3050),  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3082),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3119),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3208),  
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 3233),  
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 3268),  
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3312),  
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3401),  
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3425),  
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3491),  
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3678),  
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3836),  
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3861),  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4014),  
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4054),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (p. 219),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (p. 259),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (p. 342),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (p. 384),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (p. 411),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (p. 456),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (p. 501),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

(p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
 (p. 589), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
 (p. 619), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
 (p. 644), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
 (p. 673), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
 (p. 701), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
 (p. 774), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 878),  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 910), ac-  
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1278),  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1310),  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1341),  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1371),  
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1416),  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1445),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1478), ac-  
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1507), ac-  
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1541),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1604),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1742),  
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1776), ac-  
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1860),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1954),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2108),  
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2177),  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2206),  
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2229), ac-  
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2260),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2283),  
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller  
 (p. 2322), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller  
 (p. 2369), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller  
 (p. 2579), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller  
 (p. 2620), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
 (p. 2649), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2674),  
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2709), ac-  
 tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2758), ac-  
 tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2814),  
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2940),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3046),  
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3078),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3103),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3220), ac-  
 tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3249), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3256), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3297),  
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3389), ac-  
 tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3433), ac-  
 tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3495), ac-  
 tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3690),  
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3840),  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3869),  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4006),  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4046),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller

(p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
 (p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`  
 (p. 346), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
 (p. 388), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
 (p. 415), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
 (p. 460), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`  
 (p. 505), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
 (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`  
 (p. 593), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
 (p. 623), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
 (p. 652), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
 (p. 681), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`  
 (p. 709), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`  
 (p. 788), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 886),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 918), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1286),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1318),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1349),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1379),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1424),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1453),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1486), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1515), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1549),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1588),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1754),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1784), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1856),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1962),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2116),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2169),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2190),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2237), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2256),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2287),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
 (p. 2318), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
 (p. 2365), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
 (p. 2587), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
 (p. 2616), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
 (p. 2657), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2678),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2694), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2750), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2806),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2932),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3054),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3086),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3115),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3216), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
 (p. 3245), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
 (p. 3276), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3397), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3417), `ac-`

tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3487),  
 tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3686),  
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3824),  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3853),  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3994),  
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4058),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
 (p. 231), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
 (p. 267), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller  
 (p. 354), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
 (p. 396), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
 (p. 423), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
 (p. 468), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
 (p. 513), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
 (p. 574), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller  
 (p. 601), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
 (p. 631), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
 (p. 660), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
 (p. 685), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
 (p. 713), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller  
 (p. 795), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 890),  
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 922), ac-  
 tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1290),  
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1322),  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1353),  
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1383),  
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1428),  
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1457),  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1490), ac-  
 tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1519), ac-  
 tivemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1553),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1592),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1750),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1764), ac-  
 tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1844),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1942),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2096),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2161),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2202),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2225), ac-  
 tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2244),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2271),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2306), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2353), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2567), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2628), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2637), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2686),  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2719), ac-  
 tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2766), ac-  
 tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2802),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2924),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3058),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3090),

activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3123),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3204),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3241),  
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3272),  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3322),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3393),  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3413),  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3475),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3694),  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3844),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3865),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3998), and  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4038).

**6.301.3.5** virtual int ac-  
 tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1  
 (OpenWireFormat \* *format*, commands::DataStructure \* *command*,  
 utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [pure  
 virtual]

Tight Marshal to the given stream.

#### Parameters:

*format* - The OpenWireFormat properties  
*command* - the object to Marshal  
*bs* - boolean stream to **marshal** (p. 107) to.

#### Exceptions:

*IOException* if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller  
 (p. 215), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller  
 (p. 255), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller  
 (p. 339), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller  
 (p. 380), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller  
 (p. 407), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller  
 (p. 452), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller  
 (p. 497), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller  
 (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller  
 (p. 586), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller  
 (p. 615), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller  
 (p. 648), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller  
 (p. 677), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller  
 (p. 705), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller  
 (p. 782), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 882),  
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 914),  
 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1282),  
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1314),  
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1345),  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1375),  
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1420),

activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1449),  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1482), ac-  
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1511), ac-  
 tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1545),  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1608),  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1746),  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1780), ac-  
 tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1864),  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1958),  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2112),  
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2181),  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2210),  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2233), ac-  
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2264),  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2291),  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller  
 (p. 2326), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller  
 (p. 2373), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller  
 (p. 2583), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller  
 (p. 2624), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
 (p. 2653), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2690),  
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2715), ac-  
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2762), ac-  
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2818),  
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2944),  
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3062),  
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3094),  
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3111),  
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3212), ac-  
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller  
 (p. 3229), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller  
 (p. 3260), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3317),  
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3405), ac-  
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3421), ac-  
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3483), ac-  
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3682),  
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3829),  
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3857),  
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4010),  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4050),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller  
 (p. 223), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
 (p. 271), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
 (p. 351), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
 (p. 392), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
 (p. 419), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
 (p. 464), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
 (p. 509), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
 (p. 570), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
 (p. 598), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
 (p. 627), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
 (p. 656), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
 (p. 689), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
 (p. 717), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller

(p. 803), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 894),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 926), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1294),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1302),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1333),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1363),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1408),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1437),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1470), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1499), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1533),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1596),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1734),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1768), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1848),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1946),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2100),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2165),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2194),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2217), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2248),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2275),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`  
(p. 2314), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`  
(p. 2357), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`  
(p. 2571), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`  
(p. 2608), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
(p. 2641), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2670),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2705), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2746), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2798),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2928),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3042),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3074),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3107),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3200), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
(p. 3237), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
(p. 3264), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3385), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3429), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3479), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3698),  
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3833),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3873),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4002),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4042),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 251), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`  
(p. 335), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 376), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 403), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 448), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`



(p. 493), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
 (p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`  
 (p. 582), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
 (p. 611), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
 (p. 640), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
 (p. 669), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
 (p. 697), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`  
 (p. 768), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 874),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 906), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1274),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1306),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1337),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1367),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1412),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1441),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1474), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1503), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1537),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1600),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1738),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1772), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1852),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1950),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2104),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2173),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2198),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2221), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2252),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2279),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`  
 (p. 2310), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`  
 (p. 2361), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`  
 (p. 2575), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`  
 (p. 2612), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
 (p. 2645), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2682),  
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2700), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2754), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2810),  
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2936),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3050),  
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3082),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3119),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3208), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
 (p. 3233), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
 (p. 3268), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312),  
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3401), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3425), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3491), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3678),  
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3837),  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3861),  
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4014),  
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4054),

activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
 (p. 219), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
 (p. 259), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
 (p. 343), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
 (p. 384), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
 (p. 411), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
 (p. 456), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
 (p. 501), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
 (p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
 (p. 590), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
 (p. 619), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
 (p. 644), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
 (p. 673), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
 (p. 701), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
 (p. 775), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 878),  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 910), ac-  
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1278),  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1310),  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1341),  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1371),  
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1416),  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1445),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1478), ac-  
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1507), ac-  
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1541),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1604),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1742),  
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1776), ac-  
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1860),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1954),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2108),  
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2177),  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2206),  
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2229), ac-  
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2260),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2283),  
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller  
 (p. 2322), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller  
 (p. 2369), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller  
 (p. 2579), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller  
 (p. 2620), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
 (p. 2649), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2674),  
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2710), ac-  
 tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2758), ac-  
 tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2814),  
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2940),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3046),  
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3078),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3103),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3220), ac-  
 tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3249), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3256), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3297),  
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3389), ac-

tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3433),  
 tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3495),  
 tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3690),  
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3841),  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3869),  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4006),  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4046),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
 (p. 227), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
 (p. 263), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
 (p. 347), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
 (p. 388), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
 (p. 415), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
 (p. 460), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
 (p. 505), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
 (p. 566), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
 (p. 594), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
 (p. 623), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
 (p. 652), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
 (p. 681), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
 (p. 709), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
 (p. 789), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 886),  
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 918),  
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1286),  
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1318),  
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1349),  
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1379),  
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1424),  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1453),  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1486),  
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1515),  
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1549),  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1588),  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1754),  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1784),  
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1856),  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1962),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2116),  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2169),  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2190),  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2237),  
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2256),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2287),  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
 (p. 2318), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller  
 (p. 2365), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
 (p. 2587), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
 (p. 2616), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
 (p. 2657), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2678),  
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2695),  
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2750),  
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2806),  
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2932),  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3054),

activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3086),  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3115),  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3216),  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
 (p. 3245),  
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
 (p. 3276),  
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3307),  
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3397),  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3417),  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3487),  
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3686),  
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3825),  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3853),  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3994),  
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4058),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
 (p. 231),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
 (p. 267),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller  
 (p. 355),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
 (p. 396),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
 (p. 423),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
 (p. 468),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
 (p. 513),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
 (p. 574),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller  
 (p. 602),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
 (p. 631),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
 (p. 660),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
 (p. 685),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
 (p. 713),  
 activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller  
 (p. 796),  
 activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 890),  
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 922),  
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1290),  
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1322),  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1353),  
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1383),  
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1428),  
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1457),  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1490),  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1519),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1553),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1592),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1750),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1764),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1844),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1942),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2096),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2161),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2202),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2225),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2244),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2271),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2306),  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2353),  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2567),  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller

(p. 2628), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2637), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2686), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2720), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2766), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2802), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2924), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3058), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3090), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3123), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3204), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3241), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3272), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3393), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3413), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3475), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3694), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3845), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3865), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3998), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4038).

**6.301.3.6 virtual void `activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2`**  
**(`OpenWireFormat * format`, `commands::DataStructure * command`,  
`decaf::io::DataOutputStream * ds`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`)** [pure virtual]

Tight Marshal to the given stream.

#### Parameters:

*format* - The `OpenWireFormat` properties  
*command* - the object to Marshal  
*ds* - the `DataOutputStream` to Marshal to  
*bs* - boolean stream to `marshal` (p. 107) to.

#### Exceptions:

*IOException* if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 380), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 407), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 497), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 558), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 586), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 615), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 648), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`

(p. 677), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`  
 (p. 705), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`  
 (p. 783), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 882),  
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 914), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1282),  
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1314),  
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1345),  
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1375),  
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1420),  
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1449),  
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1482), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1511), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1546),  
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1609),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1746),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1780), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1865),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1958),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2113),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2181),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2210),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2233), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2264),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2291),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`  
 (p. 2327), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`  
 (p. 2373), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`  
 (p. 2583), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`  
 (p. 2624), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
 (p. 2653), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2690),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2716), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2762), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2818),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2945),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3062),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3094),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3111),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3212), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
 (p. 3229), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
 (p. 3260), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3318),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3405), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3421), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3483), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3682),  
`activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3829),  
`activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3857),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 4010),  
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4050),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`  
 (p. 223), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
 (p. 271), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`  
 (p. 351), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
 (p. 392), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`

(p. 419), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
 (p. 464), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`  
 (p. 509), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
 (p. 570), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`  
 (p. 598), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`  
 (p. 627), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`  
 (p. 656), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
 (p. 689), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`  
 (p. 717), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`  
 (p. 804), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 894),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 926), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1294),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1302),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1333),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1363),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1408),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1437),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1470), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1499), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1534),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1597),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1734),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1768), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1849),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1946),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2101),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2165),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2194),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2217), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2248),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2275),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`  
 (p. 2315), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`  
 (p. 2357), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`  
 (p. 2571), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`  
 (p. 2608), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
 (p. 2641), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2670),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2706), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2746), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2798),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2929),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3042),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3074),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3107),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3200), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
 (p. 3237), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
 (p. 3264), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3303),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3385), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3429), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3479), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3698),  
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3833),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3873),

activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4002),  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4042),  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 211), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 251), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
 (p. 335), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 376), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 403), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 448), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
 (p. 493), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
 (p. 582), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 (p. 611), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
 (p. 640), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
 (p. 669), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
 (p. 697), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
 (p. 769), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 874),  
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 906), ac-  
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1274),  
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1306),  
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1337),  
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1367),  
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1412),  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1441),  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1474), ac-  
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1503), ac-  
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1538),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1601),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1738),  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1772), ac-  
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1853),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1950),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2105),  
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2173),  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2198),  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2221), ac-  
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2252),  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2279),  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
 (p. 2311), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
 (p. 2361), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
 (p. 2575), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
 (p. 2612), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
 (p. 2645), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2682),  
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2701), ac-  
 tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2754), ac-  
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2810),  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2937),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3050),  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3082),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3119),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3208), ac-  
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
 (p. 3233), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller



(p. 3268), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3313),  
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3401), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3425), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3491), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3678),  
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3837),  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3861),  
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4014),  
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4054),  
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
(p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
(p. 259), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`  
(p. 343), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
(p. 384), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
(p. 411), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
(p. 456), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`  
(p. 501), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
(p. 562), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`  
(p. 590), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
(p. 619), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
(p. 644), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 673), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
(p. 701), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`  
(p. 776), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 878),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 910), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1278),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1341),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1371),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1416),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1445),  
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1478), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1507), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1542),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1605),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1742),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1776), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1861),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1954),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2109),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2177),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2206),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2229), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2260),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2283),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
(p. 2323), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
(p. 2369), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
(p. 2579), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
(p. 2620), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2649), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2674),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2711), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2758), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2814),

activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2941),  
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3046),  
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3078),  
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3103),  
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3220),  
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
 (p. 3249),  
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
 (p. 3256),  
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3298),  
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3389),  
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3433),  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3495),  
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3690),  
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3841),  
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3869),  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4006),  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4046),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
 (p. 227),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
 (p. 263),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
 (p. 347),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
 (p. 388),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
 (p. 415),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
 (p. 460),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
 (p. 505),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
 (p. 566),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
 (p. 594),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
 (p. 623),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
 (p. 652),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
 (p. 681),  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
 (p. 709),  
 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
 (p. 790),  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 886),  
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 918),  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1286),  
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1318),  
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1349),  
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1379),  
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1424),  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1453),  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1486),  
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1515),  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1550),  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1589),  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1754),  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1784),  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1857),  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1962),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2117),  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2169),  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2190),  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2237),  
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2256),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2287),  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
 (p. 2319),  
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller

(p. 2365),       activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
(p. 2587),       activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
(p. 2616), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
(p. 2657), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2678),  
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2696), ac-  
tivismq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2750), ac-  
tivismq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2806),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2933),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3054),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3086),  
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3115),  
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3216), ac-  
tivismq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
(p. 3245),       activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
(p. 3276), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3308),  
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3397), ac-  
tivismq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3417), ac-  
tivismq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3487), ac-  
tivismq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3686),  
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3825),  
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3853),  
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3994),  
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4058),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
(p. 231), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
(p. 267), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller  
(p. 355), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
(p. 396),       activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
(p. 423), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
(p. 468),       activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
(p. 513), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
(p. 574), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller  
(p. 602), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
(p. 631), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
(p. 660), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
(p. 685),       activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
(p. 713),       activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller  
(p. 797),       activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 890),  
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 922), ac-  
tivismq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1290),  
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1322),  
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1353),  
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1383),  
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1428),  
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1457),  
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1490), ac-  
tivismq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1519), ac-  
tivismq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1554),  
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1593),  
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1750),  
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1764), ac-  
tivismq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1845),  
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1942),  
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2097),

activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2161),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2202),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2225),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2244),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2271),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2307), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2353), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2567), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2628), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2637), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2686),  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2721),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2766),  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2802),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2925),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3058),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3090),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3123),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3204),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3241), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3272), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3323),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3393),  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3413),  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3475),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3694),  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3845),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3865),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3998), and  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4038).

**6.301.3.7 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal**  
 (OpenWireFormat \* *format*, commands::DataStructure \* *command*,  
 decaf::io::DataInputStream \* *dis*, utils::BooleanStream \* *bs*) throw (  
 decaf::io::IOException ) [pure virtual]

Tight Un-marhsal to the given stream.

#### Parameters:

*format* - The OpenWireFormat properties  
*command* - the object to Un-Marshal  
*dis* - the DataInputStream to Un-Marshal from  
*bs* - boolean stream to unmarshal from.

#### Exceptions:

*IOException* if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`  
 (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`  
 (p. 256), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`

(p. 340), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`  
 (p. 381), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`  
 (p. 408), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`  
 (p. 453), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`  
 (p. 498), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`  
 (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`  
 (p. 587), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`  
 (p. 616), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`  
 (p. 649), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`  
 (p. 678), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`  
 (p. 706), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`  
 (p. 784), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 883),  
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 915), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1283),  
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1315),  
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1346),  
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1376),  
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1421),  
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1450),  
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1483), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1512), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1546),  
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1609),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1747),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1781), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1865),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1959),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2113),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2182),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2211),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2234), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2265),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2292),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`  
 (p. 2327), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`  
 (p. 2374), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`  
 (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`  
 (p. 2625), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
 (p. 2654), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2691),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2716), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2763), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2819),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2945),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3063),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3095),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3112),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3213), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
 (p. 3230), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
 (p. 3261), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3319),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3406), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3422), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3484), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3683),

activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3830),  
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3858),  
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4011),  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4051),  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller  
 (p. 224), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
 (p. 272), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
 (p. 352), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
 (p. 393), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
 (p. 420), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
 (p. 465), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
 (p. 510), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
 (p. 599), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
 (p. 628), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
 (p. 657), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
 (p. 690), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
 (p. 718), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller  
 (p. 805), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 895),  
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 927), ac-  
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1295),  
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1303),  
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1334),  
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1364),  
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1409),  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1438),  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1471), ac-  
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1500), ac-  
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1534),  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1597),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1735),  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1769), ac-  
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1849),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1947),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2101),  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2166),  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2195),  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2218), ac-  
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2249),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2276),  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
 (p. 2315), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
 (p. 2358), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
 (p. 2572), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
 (p. 2609), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2642), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2671),  
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2706), ac-  
 tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2747), ac-  
 tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2799),  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2929),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3043),  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3075),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3108),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3201), ac-

tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3238), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3265), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3304),  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3386), ac-  
 tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3430), ac-  
 tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3480), ac-  
 tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3699),  
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3834),  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3874),  
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4003),  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4043),  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 212), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 252), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
 (p. 336), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 377), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 404), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 449), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
 (p. 494), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
 (p. 583), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 (p. 612), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
 (p. 641), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
 (p. 670), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
 (p. 698), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
 (p. 770), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 875),  
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 907), ac-  
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1275),  
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1307),  
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1338),  
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1368),  
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1413),  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1442),  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1475), ac-  
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1504), ac-  
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1538),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1601),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1739),  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1773), ac-  
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1853),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1951),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2105),  
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2174),  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2199),  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2222), ac-  
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2253),  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2280),  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
 (p. 2311), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
 (p. 2362), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
 (p. 2576), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
 (p. 2613), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
 (p. 2646), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2683),  
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2701), ac-

activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2755),  
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2811),  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2937),  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3051),  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3083),  
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3120),  
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3209),  
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
 (p. 3234),  
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
 (p. 3269),  
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3314),  
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3402),  
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3426),  
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3492),  
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3679),  
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3838),  
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3862),  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4015),  
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4055),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
 (p. 220),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
 (p. 260),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
 (p. 344),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
 (p. 385),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
 (p. 412),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
 (p. 457),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
 (p. 502),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
 (p. 563),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
 (p. 591),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
 (p. 620),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
 (p. 645),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
 (p. 674),  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
 (p. 702),  
 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
 (p. 777),  
 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 879),  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 911),  
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1279),  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1311),  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1342),  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1372),  
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1417),  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1446),  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1479),  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1508),  
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1542),  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1605),  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1743),  
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1777),  
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1861),  
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1955),  
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2109),  
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2178),  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2207),  
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2230),  
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2261),  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2284),



activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2323),  
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2370),  
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2580),  
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2621),  
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2650),  
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2675),  
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2711),  
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2759),  
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2815),  
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2941),  
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3047),  
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3079),  
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3104),  
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3221),  
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3250),  
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3257),  
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3299),  
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3390),  
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3434),  
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3496),  
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3691),  
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3842),  
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3870),  
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4007),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4047),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 228),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 264),  
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 348),  
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 389),  
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 416),  
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 461),  
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 506),  
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (p. 567),  
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (p. 595),  
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (p. 624),  
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (p. 653),  
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (p. 682),  
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (p. 710),  
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (p. 791),  
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 887),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 919),  
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1287),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1319),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1350),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1380),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1425),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1454),  
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1487),  
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1516),  
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1550),  
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1589),  
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1755),  
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1785),  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1857),

activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1963),  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2117),  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2170),  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2191),  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2238),  
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2257),  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2288),  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
 (p. 2319), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller  
 (p. 2366), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
 (p. 2588), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
 (p. 2617), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
 (p. 2658), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2679),  
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2696),  
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2751),  
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2807),  
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2933),  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3055),  
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3087),  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3116),  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3217),  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
 (p. 3246), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
 (p. 3277), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3309),  
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3398),  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3418),  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3488),  
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3687),  
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3826),  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3854),  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3995),  
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4059),  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
 (p. 232), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
 (p. 268), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller  
 (p. 356), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
 (p. 397), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
 (p. 424), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
 (p. 469), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
 (p. 514), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
 (p. 575), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller  
 (p. 603), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
 (p. 632), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
 (p. 661), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
 (p. 686), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
 (p. 714), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller  
 (p. 798), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 891),  
 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 923),  
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1291),  
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1323),  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1354),  
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1384),  
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1429),  
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1458),

activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1491),  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1520),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1554),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1593),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1751),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1765),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1845),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 1943),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2097),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2162),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2203),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2226),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2245),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2272),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2307), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2354), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2568), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2629), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2638), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2687),  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2721),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2767),  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2803),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2925),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3059),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3091),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3124),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3205),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3242), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3273), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3324),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3394),  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3414),  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3476),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3695),  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3846),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3866),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3999), and  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4039).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

## 6.302 activemq::commands::DataStructure Class Reference

#include <src/main/activemq/commands/DataStructure.h> Inheritance diagram for activemq::commands::DataStructure:

### Public Member Functions

- virtual **~DataStructure** ()
- virtual unsigned char **getDataStructureType** () const =0  
*Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.*
- virtual **DataStructure \* cloneDataStructure** () const =0  
*Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)=0  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const =0  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const =0  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*

### 6.302.1 Constructor & Destructor Documentation

**6.302.1.1** virtual activemq::commands::DataStructure::~~DataStructure ()  
 [inline, virtual]

### 6.302.2 Member Function Documentation

**6.302.2.1** virtual **DataStructure\*** **activemq::commands::DataStructure::cloneDataStructure** ()  
 const [pure virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 205), **activemq::commands::ActiveMQBytesMessage** (p. 236), **activemq::commands::ActiveMQDestination** (p. 324), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage**

(p. 444),      `activemq::commands::ActiveMQQueue`      (p. 484),      `ac-`  
`tivemq::commands::ActiveMQStreamMessage`      (p. 540),      `ac-`  
`tivemq::commands::ActiveMQTempDestination`      (p. 577),      `ac-`  
`tivemq::commands::ActiveMQTempQueue`      (p. 605),      `ac-`  
`tivemq::commands::ActiveMQTempTopic` (p. 634), `activemq::commands::ActiveMQTextMessage`  
(p. 664),      `activemq::commands::ActiveMQTopic`      (p. 692),      `ac-`  
`tivemq::commands::BooleanExpression` (p. 855), `activemq::commands::BrokerError`  
(p. 864), `activemq::commands::BrokerId` (p. 870), `activemq::commands::BrokerInfo`  
(p. 897),      `activemq::commands::ConnectionControl`      (p. 1268),      `ac-`  
`tivemq::commands::ConnectionError` (p. 1297), `activemq::commands::ConnectionId`  
(p. 1328),      `activemq::commands::ConnectionInfo`      (p. 1356),      `ac-`  
`tivemq::commands::ConsumerControl` (p. 1402), `activemq::commands::ConsumerId`  
(p. 1431),      `activemq::commands::ConsumerInfo`      (p. 1461),      `ac-`  
`tivemq::commands::ControlCommand` (p. 1494), `activemq::commands::DataArrayResponse`  
(p. 1529),      `activemq::commands::DataResponse`      (p. 1584),      `ac-`  
`tivemq::commands::DestinationInfo` (p. 1728), `activemq::commands::DiscoveryEvent`  
(p. 1759),      `activemq::commands::ExceptionResponse`      (p. 1840),      `ac-`  
`tivemq::commands::FlushCommand` (p. 1938), `activemq::commands::IntegerResponse`  
(p. 2092),      `activemq::commands::JournalQueueAck`      (p. 2157),      `ac-`  
`tivemq::commands::JournalTopicAck` (p. 2184), `activemq::commands::JournalTrace`  
(p. 2213),      `activemq::commands::JournalTransaction`      (p. 2240),      `ac-`  
`tivemq::commands::KeepAliveInfo` (p. 2267), `activemq::commands::LastPartialCommand`  
(p. 2301),      `activemq::commands::LocalTransactionId`      (p. 2348),      `ac-`  
`tivemq::commands::Message` (p. 2520),      `activemq::commands::MessageAck`  
(p. 2560),      `activemq::commands::MessageDispatch`      (p. 2595),      `ac-`  
`tivemq::commands::MessageDispatchNotification`      (p. 2631),      `ac-`  
`tivemq::commands::MessageId` (p. 2664),      `activemq::commands::MessagePull`  
(p. 2740),      `activemq::commands::NetworkBridgeFilter`      (p. 2794),      `ac-`  
`tivemq::commands::PartialCommand` (p. 2919), `activemq::commands::ProducerAck`  
(p. 3037),      `activemq::commands::ProducerId`      (p. 3068),      `ac-`  
`tivemq::commands::ProducerInfo` (p. 3097), `activemq::commands::RemoveInfo`  
(p. 3195),      `activemq::commands::RemoveSubscriptionInfo`      (p. 3223),      `ac-`  
`tivemq::commands::ReplayCommand` (p. 3252), `activemq::commands::Response`  
(p. 3286),      `activemq::commands::SessionId`      (p. 3380),      `ac-`  
`tivemq::commands::SessionInfo` (p. 3408), `activemq::commands::ShutdownInfo`  
(p. 3471),      `activemq::commands::SubscriptionInfo`      (p. 3672),      `ac-`  
`tivemq::commands::TransactionId` (p. 3820), `activemq::commands::TransactionInfo`  
(p. 3848),      `activemq::commands::WireFormatInfo`      (p. 3985),      and      `ac-`  
`tivemq::commands::XATransactionId` (p. 4032).

### 6.302.2.2 virtual void `activemq::commands::DataStructure::copyDataStructure` (const `DataStructure * src`) [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns:

`src` - Source Object

### 6.302.2.3 virtual bool activemq::commands::DataStructure::equals (const DataStructure \* *value*) const [pure virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if DataStructure's are Equal.

### 6.302.2.4 virtual unsigned char activemq::commands::DataStructure::getDataStructureType () const [pure virtual]

Get the **DataStructure** (p. 1660) Type as defined in CommandTypes.h.

#### Returns:

The type of the data structure

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 206), **activemq::commands::ActiveMQBytesMessage** (p. 238), **activemq::commands::ActiveMQDestination** (p. 326), **activemq::commands::ActiveMQMapMessage** (p. 366), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 445), **activemq::commands::ActiveMQQueue** (p. 485), **activemq::commands::ActiveMQStreamMessage** (p. 541), **activemq::commands::ActiveMQTempDestination** (p. 578), **activemq::commands::ActiveMQTempQueue** (p. 607), **activemq::commands::ActiveMQTempTopic** (p. 636), **activemq::commands::ActiveMQTextMessage** (p. 664), **activemq::commands::ActiveMQTopic** (p. 693), **activemq::commands::BrokerError** (p. 865), **activemq::commands::BrokerId** (p. 871), **activemq::commands::BrokerInfo** (p. 899), **activemq::commands::ConnectionControl** (p. 1269), **activemq::commands::ConnectionError** (p. 1298), **activemq::commands::ConnectionId** (p. 1329), **activemq::commands::ConnectionInfo** (p. 1357), **activemq::commands::ConsumerControl** (p. 1403), **activemq::commands::ConsumerId** (p. 1432), **activemq::commands::ConsumerInfo** (p. 1462), **activemq::commands::ControlCommand** (p. 1495), **activemq::commands::DataArrayResponse** (p. 1530), **activemq::commands::DataResponse** (p. 1585), **activemq::commands::DestinationInfo** (p. 1729), **activemq::commands::DiscoveryEvent** (p. 1760), **activemq::commands::ExceptionResponse** (p. 1840), **activemq::commands::FlushCommand** (p. 1938), **activemq::commands::IntegerResponse** (p. 2092), **activemq::commands::JournalQueueAck** (p. 2157), **activemq::commands::JournalTopicAck** (p. 2185), **activemq::commands::JournalTrace** (p. 2213), **activemq::commands::JournalTransaction** (p. 2240), **activemq::commands::KeepAliveInfo** (p. 2267), **activemq::commands::LastPartialCommand** (p. 2302), **activemq::commands::LocalTransactionId** (p. 2349), **activemq::commands::Message** (p. 2523), **activemq::commands::MessageAck** (p. 2561), **activemq::commands::MessageDispatch** (p. 2596), **activemq::commands::MessageDispatchNotification** (p. 2632), **activemq::commands::MessageId** (p. 2665), **activemq::commands::MessagePull**

(p. 2741), **activemq::commands::NetworkBridgeFilter** (p. 2794), **activemq::commands::PartialCommand** (p. 2920), **activemq::commands::ProducerAck** (p. 3037), **activemq::commands::ProducerId** (p. 3069), **activemq::commands::ProducerInfo** (p. 3098), **activemq::commands::RemoveInfo** (p. 3195), **activemq::commands::RemoveSubscriptionInfo** (p. 3224), **activemq::commands::ReplayCommand** (p. 3252), **activemq::commands::Response** (p. 3287), **activemq::commands::SessionId** (p. 3381), **activemq::commands::SessionInfo** (p. 3409), **activemq::commands::ShutdownInfo** (p. 3471), **activemq::commands::SubscriptionInfo** (p. 3673), **activemq::commands::TransactionId** (p. 3821), **activemq::commands::TransactionInfo** (p. 3849), **activemq::commands::WireFormatInfo** (p. 3986), and **activemq::commands::XATransactionId** (p. 4033).

### 6.302.2.5 virtual std::string activemq::commands::DataStructure::toString () const [pure virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 208), **activemq::commands::ActiveMQBytesMessage** (p. 244), **activemq::commands::ActiveMQDestination** (p. 330), **activemq::commands::ActiveMQMapMessage** (p. 372), **activemq::commands::ActiveMQMessage** (p. 400), **activemq::commands::ActiveMQObjectMessage** (p. 445), **activemq::commands::ActiveMQQueue** (p. 486), **activemq::commands::ActiveMQStreamMessage** (p. 547), **activemq::commands::ActiveMQTempDestination** (p. 578), **activemq::commands::ActiveMQTempQueue** (p. 607), **activemq::commands::ActiveMQTempTopic** (p. 636), **activemq::commands::ActiveMQTextMessage** (p. 666), **activemq::commands::ActiveMQTopic** (p. 694), **activemq::commands::BaseCommand** (p. 763), **activemq::commands::BaseDataStructure** (p. 833), **activemq::commands::BooleanExpression** (p. 856), **activemq::commands::BrokerId** (p. 871), **activemq::commands::BrokerInfo** (p. 901), **activemq::commands::Command** (p. 1198), **activemq::commands::ConnectionControl** (p. 1270), **activemq::commands::ConnectionError** (p. 1298), **activemq::commands::ConnectionId** (p. 1329), **activemq::commands::ConnectionInfo** (p. 1359), **activemq::commands::ConsumerControl** (p. 1404), **activemq::commands::ConsumerId** (p. 1433), **activemq::commands::ConsumerInfo** (p. 1464), **activemq::commands::ControlCommand** (p. 1495), **activemq::commands::DataArrayResponse** (p. 1530), **activemq::commands::DataResponse** (p. 1585), **activemq::commands::DestinationInfo** (p. 1730), **activemq::commands::DiscoveryEvent** (p. 1760), **activemq::commands::ExceptionResponse** (p. 1841), **activemq::commands::FlushCommand** (p. 1939), **activemq::commands::IntegerResponse** (p. 2093), **activemq::commands::JournalQueueAck** (p. 2158), **activemq::commands::JournalTopicAck** (p. 2186), **activemq::commands::JournalTrace** (p. 2214), **activemq::commands::JournalTransaction** (p. 2241), **activemq::commands::KeepAliveInfo** (p. 2268), **activemq::commands::LastPartialCommand** (p. 2302), **activemq::commands::LocalTransactionId** (p. 2350), **activemq::commands::Message** (p. 2530), **activemq::commands::MessageAck**

(p. 2563), `activemq::commands::MessageDispatch` (p. 2597), `activemq::commands::MessageDispatchNotification` (p. 2633), `activemq::commands::MessageId` (p. 2666), `activemq::commands::MessagePull` (p. 2742), `activemq::commands::NetworkBridgeFilter` (p. 2795), `activemq::commands::PartialCommand` (p. 2920), `activemq::commands::ProducerAck` (p. 3038), `activemq::commands::ProducerId` (p. 3070), `activemq::commands::ProducerInfo` (p. 3099), `activemq::commands::RemoveInfo` (p. 3196), `activemq::commands::RemoveSubscriptionInfo` (p. 3225), `activemq::commands::ReplayCommand` (p. 3253), `activemq::commands::Response` (p. 3287), `activemq::commands::SessionId` (p. 3382), `activemq::commands::SessionInfo` (p. 3409), `activemq::commands::ShutdownInfo` (p. 3472), `activemq::commands::SubscriptionInfo` (p. 3674), `activemq::commands::TransactionId` (p. 3821), `activemq::commands::TransactionInfo` (p. 3849), `activemq::commands::WireFormatInfo` (p. 3991), and `activemq::commands::XATransactionId` (p. 4034).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`



## 6.303 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

#include <src/main/decaf/util/Date.h> Inheritance diagram for decaf::util::Date:

### Public Member Functions

- **Date** ()  
*Default constructor - sets time to the current System time, rounded to the nearest millisecond.*
- **Date** (long long milliseconds)  
*Constructs the date with a given time value.*
- **Date** (const **Date** &source)  
*Copy constructor.*
- **Date** & **operator=** (const **Date** &value)  
*Assigns the value of one **Date** (p. 1665) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const  
*Gets the underlying time.*
- void **setTime** (long long milliseconds)  
*Sets the underlying time.*
- bool **after** (const **Date** &when) const  
*Determines whether or not this date falls after the specified time.*
- bool **before** (const **Date** &when) const  
*Determines whether or not this date falls before the specified time.*
- std::string **toString** () const  
*Converts this **Date** (p. 1665) object to a String of the form:.*
- virtual int **compareTo** (const **Date** &value) const  
*Compares this Data object to the one given.*
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Date** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.303.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's `java.util.Date` class.

**Since:**

1.0

### 6.303.2 Constructor & Destructor Documentation

#### 6.303.2.1 `decaf::util::Date::Date ()`

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

#### 6.303.2.2 `decaf::util::Date::Date (long long milliseconds)`

Constructs the date with a given time value.

**Parameters:**

*milliseconds* The time in milliseconds;

#### 6.303.2.3 `decaf::util::Date::Date (const Date & source)`

Copy constructor.

**Parameters:**

*source* The **Date** (p. 1665) instance to copy into this one.

#### 6.303.2.4 `virtual decaf::util::Date::~~Date ()` [virtual]

### 6.303.3 Member Function Documentation

#### 6.303.3.1 `bool decaf::util::Date::after (const Date & when) const`

Determines whether or not this date falls after the specified time.

**Parameters:**

*when* The date to compare

**Returns:**

true if this date falls after when.

#### 6.303.3.2 `bool decaf::util::Date::before (const Date & when) const`

Determines whether or not this date falls before the specified time.

**Parameters:**

*when* The date to compare

**Returns:**

true if this date falls before when.

**6.303.3.3 virtual int decaf::util::Date::compareTo (const Date & *value*) const**  
[virtual]

Compares this Date object to the one given.

**Parameters:**

*value* The **Date** (p. 1665) value to compare to this one.

**Returns:**

zero if the **Date** (p. 1665) values are equal, a value less than zero if this Date value is earlier than argument value, and a value greater than zero if this **Date** (p. 1665) object is later than the argument **Date** (p. 1665) value.

**6.303.3.4 virtual bool decaf::util::Date::equals (const Date & *value*) const**  
[virtual]**Returns:**

true if this value is considered equal to the passed value.

**6.303.3.5 long long decaf::util::Date::getTime () const**

Gets the underlying time.

**Returns:**

The underlying time value in milliseconds.

**6.303.3.6 virtual bool decaf::util::Date::operator< (const Date & *value*) const**  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.303.3.7    Date& decaf::util::Date::operator= (const Date & *value*)**

Assigns the value of one **Date** (p. 1665) object to another.

**Parameters:**

*value* The value to be copied into this **Date** (p. 1665) object.

**Returns:**

reference to this object with the newly assigned value.

**6.303.3.8    virtual bool decaf::util::Date::operator== (const Date & *value*) const**  
[virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.303.3.9    void decaf::util::Date::setTime (long long *milliseconds*)**

Sets the underlying time.

**Parameters:**

*milliseconds* The underlying time value in milliseconds.

**6.303.3.10    std::string decaf::util::Date::toString () const**

Converts this **Date** (p. 1665) object to a String of the form: . dow mon dd hh:mm:ss zzz yyyy where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits.
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.

- yyyy is the year, as four decimal digits.

**Returns:**

the String representation of the **Date** (p. 1665) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

## 6.304 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

`#include <src/main/decaf/internal/DecafRuntime.h>`Inheritance diagram for decaf::internal::DecafRuntime:

### Public Member Functions

- **DecafRuntime ()**  
*Initializes the APR Runtime for a library.*
- **virtual ~DecafRuntime ()**  
*Terminates the APR Runtime for a library.*
- **apr\_pool\_t \* getGlobalPool () const**  
*Grants access to the Global APR Pool instance that should be used when creating new threads.*

### 6.304.1 Detailed Description

Handles APR initialization and termination.

### 6.304.2 Constructor & Destructor Documentation

#### 6.304.2.1 decaf::internal::DecafRuntime::DecafRuntime ()

Initializes the APR Runtime for a library.

#### 6.304.2.2 virtual decaf::internal::DecafRuntime::~~DecafRuntime () [virtual]

Terminates the APR Runtime for a library.

### 6.304.3 Member Function Documentation

#### 6.304.3.1 apr\_pool\_t\* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/DecafRuntime.h`

## 6.305 activemq::threads::DedicatedTaskRunner Class Reference

#include <src/main/activemq/threads/DedicatedTaskRunner.h> Inheritance diagram for activemq::threads::DedicatedTaskRunner:

### Public Member Functions

- **DedicatedTaskRunner** (**Task** \*task)
- virtual **~DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)

*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*

- virtual void **shutdown** ()

*Shutdown once the task has finished and the TaskRunner's thread has exited.*

- virtual void **wakeup** ()

*Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.*

### Protected Member Functions

- virtual void **run** ()

*Run method - called by the Thread class in the context of the thread.*

### 6.305.1 Constructor & Destructor Documentation

**6.305.1.1** **activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner** (**Task** \*task)

**6.305.1.2** **virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner** () [virtual]

### 6.305.2 Member Function Documentation

**6.305.2.1** **virtual void activemq::threads::DedicatedTaskRunner::run** () [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3325).

**6.305.2.2 virtual void activemq::threads::DedicatedTaskRunner::shutdown ()**  
[virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3736).

**6.305.2.3 virtual void activemq::threads::DedicatedTaskRunner::shutdown**  
**(unsigned int *timeout*)** [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

**Parameters:**

*timeout* - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 3736).

**6.305.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup ()**  
[virtual]

Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3737).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**



## 6.306 activemq::core::policies::DefaultPrefetchPolicy Class Reference

#include <src/main/activemq/core/policies/DefaultPrefetchPolicy.h> Inheritance diagram for activemq::core::policies::DefaultPrefetchPolicy:

### Public Member Functions

- **DefaultPrefetchPolicy** ()
- virtual **~DefaultPrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int **getDurableTopicPrefetch** () const  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void **setQueuePrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue.*
- virtual int **getQueuePrefetch** () const  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void **setQueueBrowserPrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int **getQueueBrowserPrefetch** () const  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void **setTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Topic.*
- virtual int **getTopicPrefetch** () const  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int **getMaxPrefetchLimit** (int value) const  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual **PrefetchPolicy** \* **clone** () const  
*Clone the Policy and return a new pointer to that clone.*

### Static Public Attributes

- static int **MAX\_PREFETCH\_SIZE**
- static int **DEFAULT\_DURABLE\_TOPIC\_PREFETCH**
- static int **DEFAULT\_QUEUE\_PREFETCH**
- static int **DEFAULT\_QUEUE\_BROWSER\_PREFETCH**
- static int **DEFAULT\_TOPIC\_PREFETCH**

## 6.306.1 Constructor & Destructor Documentation

**6.306.1.1** `activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ()`

**6.306.1.2** `virtual  
activemq::core::policies::DefaultPrefetchPolicy::~~DefaultPrefetchPolicy  
() [virtual]`

## 6.306.2 Member Function Documentation

**6.306.2.1** `virtual PrefetchPolicy* ac-  
tivemq::core::policies::DefaultPrefetchPolicy::clone ()  
const [virtual]`

Clone the Policy and return a new pointer to that clone.

### Returns:

pointer to a new **PrefetchPolicy** (p. 2976) instance that is a clone of this one.

Implements `activemq::core::PrefetchPolicy` (p. 2977).

**6.306.2.2** `virtual int ac-  
tivemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch ()  
const [inline, virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

### Returns:

value of the number of messages to prefetch.

Implements `activemq::core::PrefetchPolicy` (p. 2978).

**6.306.2.3** `virtual int ac-  
tivemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit (int  
value) const [inline, virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

### Returns:

the allowable value for a prefetch limit, either requested or the max.

Implements `activemq::core::PrefetchPolicy` (p. 2978).

**6.306.2.4** `virtual int ac-  
tivemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch  
() const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

**Returns:**

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2978).

**6.306.2.5** `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue.

**Returns:**

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2978).

**6.306.2.6** `virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch () const [inline, virtual]`

Gets the amount of messages to prefetch for a Topic.

**Returns:**

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2978).

**6.306.2.7** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Durable Topic.

**Parameters:**

*value* The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2979).

**6.306.2.8** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch (int value) [inline, virtual]`

Sets the amount of prefetched messages for a Queue Browser.

**Parameters:**

*value* The number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2979).

**6.306.2.9** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch (int value)` [inline, virtual]

Sets the amount of prefetched messages for a Queue.

**Parameters:**

*value* The number of messages to prefetch.

Implements `activemq::core::PrefetchPolicy` (p. 2979).

**6.306.2.10** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch (int value)` [inline, virtual]

Sets the amount of prefetched messages for a Topic.

**Parameters:**

*value* The number of messages to prefetch.

Implements `activemq::core::PrefetchPolicy` (p. 2979).

### 6.306.3 Field Documentation

**6.306.3.1** `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_DURABLE_TOPIC_PREFETCH` [static]

**6.306.3.2** `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_BROWSER_PREFETCH` [static]

**6.306.3.3** `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_PREFETCH` [static]

**6.306.3.4** `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_TOPIC_PREFETCH` [static]

**6.306.3.5** `int activemq::core::policies::DefaultPrefetchPolicy::MAX_PREFETCH_SIZE` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultPrefetchPolicy.h`

## 6.307 activemq::core::policies::DefaultRedeliveryPolicy Class Reference

#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>Inheritance diagram for activemq::core::policies::DefaultRedeliveryPolicy:

### Public Member Functions

- **DefaultRedeliveryPolicy** ()
- virtual **~DefaultRedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () const
- virtual void **setBackOffMultiplier** (double value)  
*Sets the Back-Off Multiplier for Message Redelivery.*
- virtual short **getCollisionAvoidancePercent** () const
- virtual void **setCollisionAvoidancePercent** (short value)
- virtual long long **getInitialRedeliveryDelay** () const  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void **setInitialRedeliveryDelay** (long long value)  
*Sets the initial time that redelivery will be delayed.*
- virtual int **getMaximumRedeliveries** () const  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void **setMaximumRedeliveries** (int value)  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual bool **isUseCollisionAvoidance** () const
- virtual void **setUseCollisionAvoidance** (bool value)
- virtual bool **isUseExponentialBackOff** () const
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getRedeliveryDelay** (long long previousDelay)  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual **RedeliveryPolicy** \* **clone** () const  
*Create a copy of this Policy and return it.*

## 6.307.1 Constructor & Destructor Documentation

**6.307.1.1** `activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy()`

**6.307.1.2** `virtual  
activemq::core::policies::DefaultRedeliveryPolicy::~~DefaultRedeliveryPolicy()  
() [virtual]`

## 6.307.2 Member Function Documentation

**6.307.2.1** `virtual RedeliveryPolicy* ac-  
tivemq::core::policies::DefaultRedeliveryPolicy::clone ()  
const [virtual]`

Create a copy of this Policy and return it.

### Returns:

pointer to a new **RedeliveryPolicy** (p. 3177) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 3178).

**6.307.2.2** `virtual double ac-  
tivemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier ()  
const [inline, virtual]`

### Returns:

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3179).

**6.307.2.3** `virtual short ac-  
tivemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent  
() const [virtual]`

### Returns:

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 3179).

**6.307.2.4** `virtual long long ac-  
tivemq::core::policies::DefaultRedeliveryPolicy::getInitialRedeliveryDelay  
() const [inline, virtual]`

Gets the initial time that redelivery of messages is delayed.

### Returns:

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 3179).

**6.307.2.5** `virtual int activemq::core::policies::DefaultRedeliveryPolicy::getMaximumRedeliveries() const [inline, virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

**Returns:**

maximum allowed redeliveries for a message.

Implements `activemq::core::RedeliveryPolicy` (p. 3179).

**6.307.2.6** `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getRedeliveryDelay(long long previousDelay) [virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

**Parameters:**

*previousDelay* The last delay that was used between message redeliveries.

**Returns:**

the new delay to use before attempting another redelivery.

Implements `activemq::core::RedeliveryPolicy` (p. 3179).

**6.307.2.7** `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance() const [inline, virtual]`

**Returns:**

whether or not collision avoidance is enabled for this Policy.

Implements `activemq::core::RedeliveryPolicy` (p. 3180).

**6.307.2.8** `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff() const [inline, virtual]`

**Returns:**

whether or not the exponential back off option is enabled.

Implements `activemq::core::RedeliveryPolicy` (p. 3180).

**6.307.2.9** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier(double value) [inline, virtual]`

Sets the Back-Off Multiplier for Message Redelivery.

**Parameters:**

*value* The new value for the back-off multiplier.

Implements **activemq::core::RedeliveryPolicy** (p. 3180).

**6.307.2.10** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent(short value) [virtual]`

**Parameters:**

*value* The collision avoidance percentage setting.

Implements **activemq::core::RedeliveryPolicy** (p. 3180).

**6.307.2.11** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay(long long value) [inline, virtual]`

Sets the initial time that redelivery will be delayed.

**Parameters:**

*value* Time in Milliseconds to wait before starting redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3180).

**6.307.2.12** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries(int maximumRedeliveries) [inline, virtual]`

Sets the Maximum allowable redeliveries for a Message.

**Parameters:**

*maximumRedeliveries* The maximum number of times that a message will be redelivered.

Implements **activemq::core::RedeliveryPolicy** (p. 3181).

**6.307.2.13** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance(bool value) [inline, virtual]`

**Parameters:**

*value* Enable or Disable collision avoidance for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 3181).



**6.307.2.14** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff (bool value)` [inline, virtual]

**Parameters:**

*value* Enable or Disable the exponential back off multiplier option.

Implements **activemq::core::RedeliveryPolicy** (p. 3181).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultRedeliveryPolicy.h`

## 6.308 decaf::internal::net::DefaultServerSocketFactory Class Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h> Inheritance diagram for decaf::internal::net::DefaultServerSocketFactory:

### Public Member Functions

- **DefaultServerSocketFactory** ()
- virtual **~DefaultServerSocketFactory** ()
- virtual **decaf::net::ServerSocket \* createServerSocket** ()
 

Create a new **ServerSocket** (p. 3352) that is unbound.  
The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port)
 

Create a new **ServerSocket** (p. 3352) that is bound to the given port.  
The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)
 

Create a new **ServerSocket** (p. 3352) that is bound to the given port.  
The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.  
**backlog** The number of pending connect request the **ServerSocket** (p. 3352) can queue.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is *NULL* than the **ServerSocket** (p. 3352) will listen on all interfaces.

**Parameters:**

*port* The port to bind the **ServerSocket** (p. 3352) to.  
*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.  
*address* The address of the interface on the local machine to bind to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

## 6.308.1 Detailed Description

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

**Since:**

1.0

## 6.308.2 Constructor & Destructor Documentation

**6.308.2.1** decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory()  
 ()

**6.308.2.2** virtual decaf::internal::net::DefaultServerSocketFactory::~~DefaultServerSocketFactory()  
 () [virtual]

## 6.308.3 Member Function Documentation

**6.308.3.1** virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int *port*, int *backlog*, const decaf::net::InetAddress \* *address*) [virtual]

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is *NULL* than the **ServerSocket** (p. 3352) will listen on all interfaces.

**Parameters:**

*port* The port to bind the **ServerSocket** (p. 3352) to.  
*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.  
*address* The address of the interface on the local machine to bind to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

***IOException*** if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.308.3.2** `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

**Parameters:**

***port*** The port to bind the **ServerSocket** (p. 3352) to.

***backlog*** The number of pending connect request the **ServerSocket** (p. 3352) can queue.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

***IOException*** if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.308.3.3** `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket (int port) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Parameters:**

***port*** The port to bind the **ServerSocket** (p. 3352) to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

***IOException*** if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3363).

**6.308.3.4** virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ()  
[virtual]

Create a new **ServerSocket** (p. 3352) that is unbound.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

***IOException*** if the **ServerSocket** (p. 3352) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3363).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultServerSocketFactory.h**

## 6.309 decaf::internal::net::DefaultSocketFactory Class Reference

SocketFactory implementation that is used to create Sockets.

#include <src/main/decaf/internal/net/DefaultSocketFactory.h> Inheritance diagram for decaf::internal::net::DefaultSocketFactory:

### Public Member Functions

- **DefaultSocketFactory** ()
- virtual **~DefaultSocketFactory** ()
- virtual **decaf::net::Socket \* createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3503) object.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if the **Socket** (p. 3503) cannot be created.*
- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
**Parameters:**  
*host The host to connect the socket to.*  
*port The port on the remote host to connect to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.*  
***UnknownHostException** (p. 3907) if the host name is not known.*
- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
*The **Socket** (p. 3503) will be bound to the specified local address and port.*  
**Parameters:**  
*host The host to connect the socket to.*  
*port The port on the remote host to connect to.*  
*ifAddress The address on the local machine to bind the **Socket** (p. 3503) to.*  
*localPort The local port to bind the **Socket** (p. 3503) to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.*  
***UnknownHostException** (p. 3907) if the host name is not known.*

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.  
*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

**IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.  
*port* The port on the remote host to connect to.  
*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.  
*localPort* The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

**IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.

## 6.309.1 Detailed Description

SocketFactory implementation that is used to create Sockets.

**Since:**

1.0

## 6.309.2 Constructor & Destructor Documentation

**6.309.2.1** `decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ()`

**6.309.2.2** `virtual decaf::internal::net::DefaultSocketFactory::~~DefaultSocketFactory ()` [virtual]

## 6.309.3 Member Function Documentation

**6.309.3.1** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & name, int port, const decaf::net::InetAddress * ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )` [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

### Parameters:

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

### Returns:

a new **Socket** (p. 3503) object, caller must free this object when done.

### Exceptions:

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

**6.309.3.2** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket (const std::string & name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )` [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

### Parameters:

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

### Returns:

a new **Socket** (p. 3503) object, caller must free this object when done.



**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

**6.309.3.3** virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*, const decaf::net::InetAddress \* *ifAddress*, int *localPort*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

The **Socket** (p. 3503) will be bound to the specified local address and port.

**Parameters:**

***host*** The host to connect the socket to.

***port*** The port on the remote host to connect to.

***ifAddress*** The address on the local machine to bind the **Socket** (p. 3503) to.

***localPort*** The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.309.3.4** virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

***host*** The host to connect the socket to.

***port*** The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.309.3.5** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket ()  
throw ( decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3503) object.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if the **Socket** (p. 3503) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3527).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/DefaultSocketFactory.h`

## 6.310 decaf::internal::net::ssl::DefaultSSLContext Class Reference

Default SSLContext manager for the Decaf library.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>
```

### Public Member Functions

- virtual `~DefaultSSLContext ()`

### Static Public Member Functions

- static `decaf::net::ssl::SSLContext * getContext ()`

### Protected Member Functions

- `DefaultSSLContext ()`

#### 6.310.1 Detailed Description

Default SSLContext manager for the Decaf library. If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled an preferred.

Since:

1.0

#### 6.310.2 Constructor & Destructor Documentation

**6.310.2.1** `decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext ()`  
[protected]

**6.310.2.2** `virtual decaf::internal::net::ssl::DefaultSSLContext::~~DefaultSSLContext ()` [virtual]

#### 6.310.3 Member Function Documentation

**6.310.3.1** `static decaf::net::ssl::SSLContext* decaf::internal::net::ssl::DefaultSSLContext::getContext ()`  
[static]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLContext.h`

## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

#include <src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h> Inheritance diagram for decaf::internal::net::ssl::DefaultSSLServerSocketFactory:

### Public Member Functions

- **DefaultSSLServerSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket \* createServerSocket** ()
 

*Create a new **ServerSocket** (p. 3352) that is unbound.*  
*The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.*

**Returns:**

*new **ServerSocket** (p. 3352) pointer that is owned by the caller.*

**Exceptions:**

*IOException if the **ServerSocket** (p. 3352) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port)
 

*Create a new **ServerSocket** (p. 3352) that is bound to the given port.*  
*The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.*

**Parameters:**

*port The port to bind the **ServerSocket** (p. 3352) to.*

**Returns:**

*new **ServerSocket** (p. 3352) pointer that is owned by the caller.*

**Exceptions:**

*IOException if the **ServerSocket** (p. 3352) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)
 

*Create a new **ServerSocket** (p. 3352) that is bound to the given port.*  
*The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.*

**Parameters:**

*port The port to bind the **ServerSocket** (p. 3352) to.*  
*backlog The number of pending connect request the **ServerSocket** (p. 3352) can queue.*

**Returns:**

*new **ServerSocket** (p. 3352) pointer that is owned by the caller.*

**Exceptions:**

*IOException if the **ServerSocket** (p. 3352) cannot be created for some reason.*
- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is **NULL** than the **ServerSocket** (p. 3352) will listen on all interfaces.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.  
**backlog** The number of pending connect request the **ServerSocket** (p. 3352) can queue.  
**address** The address of the interface on the local machine to bind to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

**getSupportedCipherSuites()** (p. 3561)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

**getDefaultCipherSuites()** (p. 3561)

## 6.311.1 Detailed Description

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

**Since:**

1.0

## 6.311.2 Constructor & Destructor Documentation

**6.311.2.1** `decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory(const std::string & errorMessage)`

**6.311.2.2** `virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory() [virtual]`

## 6.311.3 Member Function Documentation

**6.311.3.1** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket(int port, int backlog, const decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3352) will listen on all interfaces.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

*address* The address of the interface on the local machine to bind to.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.311.3.2** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket(int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.311.3.3 virtual decaf::net::ServerSocket\* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket(int port) [virtual]**

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Parameters:**

*port* The port to bind the **ServerSocket** (p. 3352) to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3363).

**6.311.3.4 virtual decaf::net::ServerSocket\* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket() [virtual]**

Create a new **ServerSocket** (p. 3352) that is unbound.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3363).

**6.311.3.5 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getDefaultCipherSuites() [virtual]**

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

`getSupportedCipherSuites()` (p. 3561)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3561).

**6.311.3.6** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getSupportedCipherSuites()` [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

`getDefaultCipherSuites()` (p. 3561)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3561).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h`



## 6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

#include <src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h>Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

### Public Member Functions

- **DefaultSSLSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3503) object.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if the **Socket** (p. 3503) cannot be created.*
- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
**Parameters:**  
***host** The host to connect the socket to.  
**port** The port on the remote host to connect to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.*
- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
*The **Socket** (p. 3503) will be bound to the specified local address and port.*  
**Parameters:**  
***host** The host to connect the socket to.  
**port** The port on the remote host to connect to.  
**ifAddress** The address on the local machine to bind the **Socket** (p. 3503) to.  
**localPort** The local port to bind the **Socket** (p. 3503) to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.*

*UnknownHostException* (p. 3907) if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*

**Parameters:**

***host** The host name or IP address to connect the socket to.  
**port** The port on the remote host to connect to.*

**Returns:**

*a new **Socket** (p. 3503) object, caller must free this object when done.*

**Exceptions:**

***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.*

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const decaf::net::InetAddress \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*

**Parameters:**

***host** The host name or IP address to connect the socket to.  
**port** The port on the remote host to connect to.  
**ifAddress** The address on the local machine to bind the **Socket** (p. 3503) to.  
**localPort** The local port to bind the **Socket** (p. 3503) to.*

**Returns:**

*a new **Socket** (p. 3503) object, caller must free this object when done.*

**Exceptions:**

***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

*Returns the list of cipher suites which are enabled by default.*

*Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).*

**Returns:**

*an STL vector containing the list of cipher suites enabled by default.*

**See also:**

***getSupportedCipherSuites()** (p. 3573)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*

*Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.*

**Returns:**

*an STL vector containing the list of supported cipher suites.*

**See also:**

***getDefaultCipherSuites()** (p. 3573)*

- virtual **decaf::net::Socket \* createSocket** (**decaf::net::Socket \*socket**, **std::string host**, **int port**, **bool autoClose**)

Returns a socket layered over an existing socket connected to the named host, at the given port. This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters:**

*socket* The existing socket to layer over.

*host* The server host the original **Socket** (p. 3503) is connected to.

*port* The server port the original **Socket** (p. 3503) is connected to.

*autoClose* Should the layered over **Socket** (p. 3503) be closed when the topmost socket is closed.

**Returns:**

a new **Socket** (p. 3503) instance that wraps the given **Socket** (p. 3503).

**Exceptions:**

**IOException** if an I/O exception occurs while performing this operation.

**UnknownHostException** (p. 3907) if the host is unknown.

### 6.312.1 Detailed Description

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

Since:

1.0

### 6.312.2 Constructor & Destructor Documentation

**6.312.2.1** **decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory** (**const std::string & errorMessage**)

**6.312.2.2** **virtual**  
**decaf::internal::net::ssl::DefaultSSLSocketFactory::~DefaultSSLSocketFactory**  
( ) [virtual]

### 6.312.3 Member Function Documentation

**6.312.3.1** **virtual decaf::net::Socket\* de-**  
**caf::internal::net::ssl::DefaultSSLSocketFactory::createSocket**  
(**decaf::net::Socket \* socket**, **std::string host**, **int port**, **bool autoClose**)  
[virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters:**

*socket* The existing socket to layer over.

*host* The server host the original **Socket** (p. 3503) is connected to.

*port* The server port the original **Socket** (p. 3503) is connected to.

*autoClose* Should the layered over **Socket** (p. 3503) be closed when the topmost socket is closed.

#### Returns:

a new **Socket** (p. 3503) instance that wraps the given **Socket** (p. 3503).

#### Exceptions:

*IOException* if an I/O exception occurs while performing this operation.

*UnknownHostException* (p. 3907) if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3572).

**6.312.3.2** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & name, int port, const decaf::net::InetAddress * ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]`

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

#### Parameters:

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

#### Returns:

a new **Socket** (p. 3503) object, caller must free this object when done.

#### Exceptions:

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

**6.312.3.3** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const std::string & name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]`

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

**6.312.3.4** virtual decaf::net::Socket\* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*, const decaf::net::InetAddress \* *ifAddress*, int *localPort*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

The **Socket** (p. 3503) will be bound to the specified local address and port.

**Parameters:**

*host* The host to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.312.3.5** virtual decaf::net::Socket\* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host to connect the socket to.

*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.312.3.6** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket () throw (decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3503) object.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if the **Socket** (p. 3503) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3527).

**6.312.3.7** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getDefaultCipherSuites () [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

**getSupportedCipherSuites()** (p. 3573)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3573).

### 6.312.3.8 virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getSupportedCipherSuites() [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

#### Returns:

an STL vector containing the list of supported cipher suites.

#### See also:

[getDefaultCipherSuites\(\)](#) (p. 3573)

Implements [decaf::net::ssl::SSLSocketFactory](#) (p. 3573).

The documentation for this class was generated from the following file:

- [src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h](#)

## 6.313 activemq::transport::DefaultTransportListener Class Reference

#include <src/main/activemq/transport/DefaultTransportListener.h> Inheritance diagram for activemq::transport::DefaultTransportListener:

### Public Member Functions

- virtual **~DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command *AMQCPP\_UNUSED*)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex *AMQCPP\_UNUSED*)  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **transportInterrupted** ()  
*The **transport** (p. 97) has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The **transport** (p. 97) has resumed after an interruption.*

### 6.313.1 Constructor & Destructor Documentation

- 6.313.1.1** virtual **activemq::transport::DefaultTransportListener::~~DefaultTransportListener** () [inline, virtual]

### 6.313.2 Member Function Documentation

- 6.313.2.1** virtual void **activemq::transport::DefaultTransportListener::onCommand** (const **Pointer**< **Command** > &command *AMQCPP\_UNUSED*) [inline, virtual]

Event handler for the receipt of a command. The **transport** (p. 97) passes off all received **commands** (p. 87) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3883) deletes the command upon receipt.

#### Parameters:

*command* the received command object.

- 6.313.2.2** virtual void **activemq::transport::DefaultTransportListener::onException** (const **decaf::lang::Exception** &ex *AMQCPP\_UNUSED*) [inline, virtual]

Event handler for an exception from a command **transport** (p. 97).



**Parameters:**

*ex* The exception.

**6.313.2.3 virtual void activemq::transport::DefaultTransportListener::transportInterrupted ()**  
[inline, virtual]

The **transport** (p.97) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p.3901).

**6.313.2.4 virtual void activemq::transport::DefaultTransportListener::transportResumed ()**  
[inline, virtual]

The **transport** (p.97) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p.3901).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

## 6.314 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see [specification](#)).

```
#include <src/main/decaf/util/zip/Deflater.h>
```

### Public Member Functions

- **Deflater** (int level, bool nowrap=false)  
*Creates a new compressor using the specified compression level.*
- **Deflater** ()  
*Creates a new compressor with the default compression level.*
- virtual **~Deflater** ()
- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setStrategy** (int strategy) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the compression strategy to the specified value.*
- void **setLevel** (int level) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Sets the compression level to the specified value.*

- bool **needsInput** () const
- void **finish** ()

*When called, indicates that compression should end with the current contents of the input buffer.*

- bool **finished** () const
- int **deflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Fills specified buffer with compressed data.*

- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Fills specified buffer with compressed data.*

- int **deflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )

*Fills specified buffer with compressed data.*

- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )

*Resets deflater so that a new set of input data can be processed.*

- void **end** ()

*Closes the compressor and discards any unprocessed input.*

## Static Public Attributes

- static const int **BEST\_SPEED**  
*Compression level for fastest compression.*
- static const int **BEST\_COMPRESSION**  
*Compression level for best compression.*
- static const int **DEFAULT\_COMPRESSION**  
*Default compression level.*
- static const int **DEFLATED**  
*Compression method for the deflate algorithm (the only one currently supported).*
- static const int **NO\_COMPRESSION**  
*Compression level for no compression.*

- static const int **FILTERED**

*Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.*

- static const int **HUFFMAN\_ONLY**

*Compression strategy for Huffman coding only.*

- static const int **DEFAULT\_STRATEGY**

*Default compression strategy.*

### 6.314.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see [specification](#)). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 1716) and its descendants.

The typical usage of a **Deflater** (p. 1706) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 1716).

See also:

**DeflaterOutputStream** (p. 1716)

**Inflater** (p. 2027)

Since:

1.0

### 6.314.2 Constructor & Destructor Documentation

#### 6.314.2.1 `decaf::util::zip::Deflater::Deflater (int level, bool nowrap = false)`

Creates a new compressor using the specified compression level. If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

Parameters:

*level* The compression level to use (0-9).

*nowrap* If true uses GZip compatible compression (defaults to false).

#### 6.314.2.2 `decaf::util::zip::Deflater::Deflater ()`

Creates a new compressor with the default compression level. Compressed data will be generated in ZLIB format.

**6.314.2.3** virtual decaf::util::zip::Deflater::~~Deflater () [virtual]

## 6.314.3 Member Function Documentation

**6.314.3.1** int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer*) throw ( decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data. Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1711) should be called in order to determine if more input data is required.

### Parameters:

*buffer* The Buffer to write the compressed data to.

### Returns:

the actual number of bytes of compressed data.

### Exceptions:

*IllegalStateException* if in the end state.

**6.314.3.2** int decaf::util::zip::Deflater::deflate (std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data. Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1711) should be called in order to determine if more input data is required.

### Parameters:

*buffer* The Buffer to write the compressed data to.

*offset* The position in the Buffer to start writing at.

*length* The maximum number of byte of data to write.

### Returns:

the actual number of bytes of compressed data.

### Exceptions:

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*IllegalStateException* if in the end state.

**6.314.3.3** int decaf::util::zip::Deflater::deflate (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data. Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1711) should be called in order to determine if more input data is required.

**Parameters:**

*buffer* The Buffer to write the compressed data to.  
*size* The size of the passed buffer.  
*offset* The position in the Buffer to start writing at.  
*length* The maximum number of byte of data to write.

**Returns:**

the actual number of bytes of compressed data.

**Exceptions:**

*NullPointerException* if buffer is NULL.  
*IndexOutOfBoundsException* if the offset + length > size of the buffer.  
*IllegalStateException* if in the end state.

**6.314.3.4 void decaf::util::zip::Deflater::end ()**

Closes the compressor and discards any unprocessed input. This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p. 1706) object is undefined.

**6.314.3.5 void decaf::util::zip::Deflater::finish ()**

When called, indicates that compression should end with the current contents of the input buffer.

**6.314.3.6 bool decaf::util::zip::Deflater::finished () const****Returns:**

true if the end of the compressed data output stream has been reached.

**6.314.3.7 long long decaf::util::zip::Deflater::getAdler () const throw ( decaf::lang::exceptions::IllegalStateException )****Returns:**

the ADLER-32 value of the uncompressed data.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.8 long long decaf::util::zip::Deflater::getBytesRead () const throw ( decaf::lang::exceptions::IllegalStateException )****Returns:**

the total number of uncompressed bytes input so far.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.9** long long decaf::util::zip::Deflater::getBytesWritten () const throw ( decaf::lang::exceptions::IllegalStateException )

**Returns:**

the total number of compressed bytes output so far.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.10** bool decaf::util::zip::Deflater::needsInput () const

**Returns:**

true if the input data buffer is empty and **setInput()** (p. 1713) should be called in order to provide more input

**6.314.3.11** void decaf::util::zip::Deflater::reset () throw ( decaf::lang::exceptions::IllegalStateException )

Resets deflater so that a new set of input data can be processed. Keeps current compression level and strategy settings.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.12** void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & *buffer*) throw ( decaf::lang::exceptions::IllegalStateException )

Sets preset dictionary for compression. A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2031), **Inflater.getAdler()** (p. 2029) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters:**

*buffer* The buffer containing the preset dictionary.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.13** `void decaf::util::zip::Deflater::setDictionary (const std::vector< unsigned char > & buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )`

Sets preset dictionary for compression. A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2031), **Inflater.getAdler()** (p. 2029) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters:**

*buffer* The buffer containing the preset dictionary.  
*offset* The position in the Buffer to start reading from.  
*length* The number of bytes to read from the input buffer.

**Exceptions:**

*IndexOutOfBoundsException* if the offset + length > size of the buffer.  
*IllegalStateException* if in the end state.

**6.314.3.14** `void decaf::util::zip::Deflater::setDictionary (const unsigned char * buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )`

Sets preset dictionary for compression. A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2031), **Inflater.getAdler()** (p. 2029) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters:**

*buffer* The buffer containing the preset dictionary.  
*size* The size of the passed dictionary buffer.  
*offset* The position in the Buffer to start reading from.  
*length* The number of bytes to read from the input buffer.

**Exceptions:**

*NullPointerException* if buffer is NULL.  
*IndexOutOfBoundsException* if the offset + length > size of the buffer.  
*IllegalStateException* if in the end state.

**6.314.3.15** `void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & buffer) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets input data for compression. This should be called whenever **needsInput()** (p. 1711) returns true indicating that more input data is required.



**Parameters:**

*buffer* The Buffer to read in for compression.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.314.3.16** void decaf::util::zip::Deflater::setInput (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for compression. This should be called whenever **needsInput()** (p. 1711) returns true indicating that more input data is required.

**Parameters:**

*buffer* The Buffer to read in for compression.

*offset* The position in the Buffer to start reading from.

*length* The number of bytes to read from the input buffer.

**Exceptions:**

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*IllegalStateException* if in the end state.

**6.314.3.17** void decaf::util::zip::Deflater::setInput (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for compression. This should be called whenever **needsInput()** (p. 1711) returns true indicating that more input data is required.

**Parameters:**

*buffer* The Buffer to read in for compression.

*size* The size in bytes of the buffer passed.

*offset* The position in the Buffer to start reading from.

*length* The number of bytes to read from the input buffer.

**Exceptions:**

*NullPointerException* if buffer is NULL.

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*IllegalStateException* if in the end state.

**6.314.3.18**    `void decaf::util::zip::Deflater::setLevel (int level) throw  
                  ( decaf::lang::exceptions::IllegalArgumentException,  
                  decaf::lang::exceptions::IllegalStateException )`

Sets the compression level to the specified value.

**Parameters:**

*level*    The new Compression level to use.

**Exceptions:**

*IllegalArgumentException*    if the level value is invalid.

*IllegalStateException*    if in the end state.

**6.314.3.19**    `void decaf::util::zip::Deflater::setStrategy (int strategy)  
                  throw ( decaf::lang::exceptions::IllegalArgumentException,  
                  decaf::lang::exceptions::IllegalStateException )`

Sets the compression strategy to the specified value.

**Parameters:**

*strategy*    The new Compression strategy to use.

**Exceptions:**

*IllegalArgumentException*    if the strategy value is invalid.

*IllegalStateException*    if in the end state.

## 6.314.4    Field Documentation

**6.314.4.1**    `const int decaf::util::zip::Deflater::BEST_COMPRESSION    [static]`

Compression level for best compression.

**6.314.4.2**    `const int decaf::util::zip::Deflater::BEST_SPEED    [static]`

Compression level for fastest compression.

**6.314.4.3**    `const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION  
                  [static]`

Default compression level.

**6.314.4.4**    `const int decaf::util::zip::Deflater::DEFAULT_STRATEGY    [static]`

Default compression strategy.

**6.314.4.5** `const int decaf::util::zip::Deflater::DEFLATED` [static]

Compression method for the deflate algorithm (the only one currently supported).

**6.314.4.6** `const int decaf::util::zip::Deflater::FILTERED` [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution. Forces more Huffman coding and less string matching.

**6.314.4.7** `const int decaf::util::zip::Deflater::HUFFMAN_ONLY` [static]

Compression strategy for Huffman coding only.

**6.314.4.8** `const int decaf::util::zip::Deflater::NO_COMPRESSION` [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Deflater.h`

## 6.315 decaf::util::zip::DeflaterOutputStream Class Reference

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

#include <src/main/decaf/util/zip/DeflaterOutputStream.h> Inheritance diagram for decaf::util::zip::DeflaterOutputStream:

### Public Member Functions

- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, bool own=false)

*Creates a new DeflateOutputStream with a Default **Deflater** (p. 1706) and buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, Deflater \*deflater, bool own=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1706) and a default buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, Deflater \*deflater, int bufferSize, bool own=false)

*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1706) and specified buffer size.*

- virtual ~**DeflaterOutputStream** ()
- virtual void **finish** () throw ( decaf::io::IOException )

*Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

**Exceptions:**

***IOException** (p. 2142) if an error occurs while closing.*

*The default implementation of this method does nothing.*

*The close method of **FilterOutputStream** (p. 1900) calls its flush method, and then calls the close method of its underlying output stream.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **deflate** () throw ( decaf::io::IOException )

*Writes a buffers worth of compressed data to the wrapped OutputStream.*

## Protected Attributes

- **Deflater \* deflater**  
*The **Deflater** (p. 1706) for this stream.*
- `std::vector< unsigned char > buf`  
*The **Buffer** to use for.*
- `bool ownDeflater`
- `bool isDone`

## Static Protected Attributes

- `static const std::size_t DEFAULT_BUFFER_SIZE`

### 6.315.1 Detailed Description

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

**Since:**

1.0

### 6.315.2 Constructor & Destructor Documentation

#### 6.315.2.1 decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream \* *outputStream*, bool *own* = false)

Creates a new DeflateOutputStream with a Default **Deflater** (p. 1706) and buffer size.

**Parameters:**

*outputStream* The OutputStream instance to wrap.

*own* Should this filter take ownership of the OutputStream pointer (default is false).

#### 6.315.2.2 decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (decaf::io::OutputStream \* *outputStream*, Deflater \* *deflater*, bool *own* = false)

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1706) and a default buffer size. When the user supplied a **Deflater** (p. 1706) instance the DeflaterOutpotStream does not take ownership of the **Deflater** (p. 1706) pointer, the caller is still responsible for deleting the **Deflater** (p. 1706).

**Parameters:**

*outputStream* The OutputStream instance to wrap.

*deflater* The user supplied **Deflater** (p. 1706) to use for compression. (

*own* Should this filter take ownership of the OutputStream pointer (default is false).

**Exceptions:**

*NullPointerException* if the **Deflater** (p.1706) given is NULL.

**6.315.2.3** `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream`  
 (`decaf::io::OutputStream * outputStream`, `Deflater * deflater`, `int bufferSize`, `bool own = false`)

Creates a new `DeflaterOutputStream` with a user supplied **Deflater** (p.1706) and specified buffer size. When the user supplied a **Deflater** (p.1706) instance the `DeflaterOutputStream` does not take ownership of the **Deflater** (p.1706) pointer, the caller is still responsible for deleting the **Deflater** (p.1706).

**Parameters:**

*outputStream* The `OutputStream` instance to wrap.

*deflater* The user supplied **Deflater** (p.1706) to use for compression.

*bufferSize* The size of the input buffer.

*own* Should this filter take ownership of the `OutputStream` pointer (default is false).

**Exceptions:**

*NullPointerException* if the **Deflater** (p.1706) given is NULL.

*IllegalArgumentException* if `bufferSize` is 0.

**6.315.2.4** `virtual decaf::util::zip::DeflaterOutputStream::~~DeflaterOutputStream ()`  
 [virtual]

**6.315.3 Member Function Documentation**

**6.315.3.1** `virtual void decaf::util::zip::DeflaterOutputStream::close () throw (`  
`decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling `close`.

**Exceptions:**

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing.

The `close` method of **FilterOutputStream** (p.1900) calls its `flush` method, and then calls the `close` method of its underlying output stream. Finishes writing any remaining data to the `OutputStream` then closes the stream.

Reimplemented from `decaf::io::FilterOutputStream` (p.1901).

**6.315.3.2** `virtual void decaf::util::zip::DeflaterOutputStream::deflate () throw (`  
`decaf::io::IOException )` [protected, virtual]

Writes a buffers worth of compressed data to the wrapped `OutputStream`.

**6.315.3.3** virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.315.3.4** virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte (unsigned char *value*) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.315.3.5** virtual void decaf::util::zip::DeflaterOutputStream::finish () throw ( decaf::io::IOException ) [virtual]

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

#### Exceptions:

*IOException* if an I/O error occurs.

### 6.315.4 Field Documentation

**6.315.4.1** std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf [protected]

The Buffer to use for.

**6.315.4.2** const std::size\_t decaf::util::zip::DeflaterOutputStream::DEFAULT\_BUFFER\_SIZE [static, protected]

**6.315.4.3** Deflater\* decaf::util::zip::DeflaterOutputStream::deflater [protected]

The Deflater (p. 1706) for this stream.

**6.315.4.4** bool decaf::util::zip::DeflaterOutputStream::isDone [protected]

**6.315.4.5** bool decaf::util::zip::DeflaterOutputStream::ownDeflater [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/DeflaterOutputStream.h

## 6.316 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

#include <src/main/decaf/util/concurrent/Delayed.h> Inheritance diagram for decaf::util::concurrent::Delayed:

### Public Member Functions

- virtual `~Delayed()`
- virtual long long `getDelay` (const `TimeUnit` &unit)=0

*Returns the remaining delay associated with this object, in the given time unit.*

### 6.316.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a `Comparable` methods that provides an ordering consistent with its `getDelay` method.

### 6.316.2 Constructor & Destructor Documentation

**6.316.2.1** virtual `decaf::util::concurrent::Delayed::~~Delayed()` [`inline`, `virtual`]

### 6.316.3 Member Function Documentation

**6.316.3.1** virtual long long `decaf::util::concurrent::Delayed::getDelay` (const `TimeUnit` & *unit*) [`pure virtual`]

Returns the remaining delay associated with this object, in the given time unit.

#### Parameters:

*unit* The time unit

#### Returns:

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Delayed.h`



## 6.317 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

### Public Types

- enum **DELIVERY\_MODE** { **PERSISTENT** = 0, **NON\_PERSISTENT** = 1 }

*Enumeration values for **Message** (p. 2534) Delivery Mode.*

### Public Member Functions

- virtual **~DeliveryMode** ()

#### 6.317.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a **cms::Message** (p. 2534) it can mark the **Message** (p. 2534) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2534) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2534) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2534) throughput.

The **DeliveryMode** (p. 1721) covers only the transport of the **Message** (p. 2534) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2534) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2534) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 2534) consumer allows for it.

**Since:**

1.0

#### 6.317.2 Member Enumeration Documentation

##### 6.317.2.1 enum cms::DeliveryMode::DELIVERY\_MODE

Enumeration values for **Message** (p. 2534) Delivery Mode.

**Enumerator:**

**PERSISTENT**

**NON\_PERSISTENT**

### 6.317.3 Constructor & Destructor Documentation

#### 6.317.3.1 `virtual cms::DeliveryMode::~~DeliveryMode ()` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/DeliveryMode.h`

## 6.318 cms::Destination Class Reference

A **Destination** (p. 1723) object encapsulates a provider-specific address.

#include <src/main/cms/Destination.h> Inheritance diagram for cms::Destination:

### Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY\_TOPIC**, **TEMPORARY\_QUEUE** }

### Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0  
*Retrieve the **Destination** (p. 1723) Type for this **Destination** (p. 1723).*
- virtual **cms::Destination \* clone** () const =0  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)=0  
*Copies the contents of the given **Destination** (p. 1723) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0  
*Retrieve any properties that might be part of the destination that was specified.*

### 6.318.1 Detailed Description

A **Destination** (p. 1723) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1723) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1723) address.

All CMS **Destination** (p. 1723) objects support concurrent use.

Since:

1.0

### 6.318.2 Member Enumeration Documentation

#### 6.318.2.1 enum cms::Destination::DestinationType

Enumerator:

**TOPIC**  
**QUEUE**  
**TEMPORARY\_TOPIC**  
**TEMPORARY\_QUEUE**

### 6.318.3 Constructor & Destructor Documentation

**6.318.3.1** `virtual cms::Destination::~~Destination () [inline, virtual]`

### 6.318.4 Member Function Documentation

**6.318.4.1** `virtual cms::Destination* cms::Destination::clone () const [pure virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

#### Returns:

cloned copy of this object

Implemented in `activemq::commands::ActiveMQQueue` (p. 484),  
`activemq::commands::ActiveMQTempQueue` (p. 605), `ac-`  
`tivemq::commands::ActiveMQTempTopic` (p. 634), and `ac-`  
`tivemq::commands::ActiveMQTopic` (p. 692).

**6.318.4.2** `virtual void cms::Destination::copy (const cms::Destination & source) [pure virtual]`

Copies the contents of the given **Destination** (p. 1723) object to this one.

#### Parameters:

*source* The source **Destination** (p. 1723) object.

**6.318.4.3** `virtual const CMSProperties& cms::Destination::getCMSProperties () const [pure virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns:

A {const} reference to a **CMSProperties** (p. 1165) object.

Implemented in `activemq::commands::ActiveMQQueue` (p. 485),  
`activemq::commands::ActiveMQTempQueue` (p. 606), `ac-`  
`tivemq::commands::ActiveMQTempTopic` (p. 635), and `ac-`  
`tivemq::commands::ActiveMQTopic` (p. 693).

**6.318.4.4** `virtual DestinationType cms::Destination::getDestinationType () const [pure virtual]`

Retrieve the **Destination** (p. 1723) Type for this **Destination** (p. 1723).

#### Returns:

The **Destination** (p. 1723) Type

Implemented in `activemq::commands::ActiveMQQueue` (p. 485),  
`activemq::commands::ActiveMQTempQueue` (p. 607), `ac-`  
`tivemq::commands::ActiveMQTempTopic` (p. 636), and `ac-`  
`tivemq::commands::ActiveMQTopic` (p. 693).

The documentation for this class was generated from the following file:

- `src/main/cms/Destination.h`

## 6.319 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

### Static Public Attributes

- static const std::string **ANY\_CHILD**
- static const std::string **ANY\_DESCENDENT**

### 6.319.1 Field Documentation

**6.319.1.1** const std::string  
activemq::commands::ActiveMQDestination::DestinationFilter::ANY\_  
CHILD [static]

**6.319.1.2** const std::string  
activemq::commands::ActiveMQDestination::DestinationFilter::ANY\_  
DESCENDENT [static]

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**ActiveMQDestination.h**

## 6.320 activemq::commands::DestinationInfo Class Reference

#include <src/main/activemq/commands/DestinationInfo.h> Inheritance diagram for activemq::commands::DestinationInfo:

### Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DestinationInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_DESTINATIONINFO** = 8

## Protected Attributes

- **Pointer< ConnectionId > connectionId**
- **Pointer< ActiveMQDestination > destination**
- unsigned char **operationType**
- long long **timeout**
- **std::vector< decaf::lang::Pointer< BrokerId > > brokerPath**

## 6.320.1 Constructor & Destructor Documentation

**6.320.1.1** **activemq::commands::DestinationInfo::DestinationInfo ()**

**6.320.1.2** **virtual activemq::commands::DestinationInfo::~~DestinationInfo ()**  
[virtual]

## 6.320.2 Member Function Documentation

**6.320.2.1** **virtual DestinationInfo\* activemq::commands::DestinationInfo::cloneDataStructure ()**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.320.2.2** **virtual void activemq::commands::DestinationInfo::copyDataStructure (const DataStructure \* *src*)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.320.2.3** **virtual bool activemq::commands::DestinationInfo::equals (const DataStructure \* *value*) const** [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.



**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.320.2.4** `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () [virtual]`
- 6.320.2.5** `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const [virtual]`
- 6.320.2.6** `virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () [virtual]`
- 6.320.2.7** `virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const [virtual]`
- 6.320.2.8** `virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.320.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ()`  
[virtual]
- 6.320.2.10 `virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const`  
[virtual]
- 6.320.2.11 `virtual unsigned char activemq::commands::DestinationInfo::getOperationType ()`  
const [virtual]
- 6.320.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout ()`  
const [virtual]
- 6.320.2.13 `virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`  
[virtual]
- 6.320.2.14 `virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.320.2.15 `virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.320.2.16 `virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char operationType)` [virtual]
- 6.320.2.17 `virtual void activemq::commands::DestinationInfo::setTimeout (long long timeout)` [virtual]
- 6.320.2.18 `virtual std::string activemq::commands::DestinationInfo::toString ()`  
const [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

- 6.320.2.19 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1198).

### 6.320.3 Field Documentation

- 6.320.3.1** `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::DestinationInfo::brokerPath` [protected]
- 6.320.3.2** `Pointer<ConnectionId>` `activemq::commands::DestinationInfo::connectionId`  
[protected]
- 6.320.3.3** `Pointer<ActiveMQDestination>` `activemq::commands::DestinationInfo::destination`  
[protected]
- 6.320.3.4** `const unsigned char` `activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8` [static]
- 6.320.3.5** `unsigned char` `activemq::commands::DestinationInfo::operationType`  
[protected]
- 6.320.3.6** `long long` `activemq::commands::DestinationInfo::timeout` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DestinationInfo.h`

## 6.321 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1732).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.321.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1732).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.321.2 Constructor & Destructor Documentation

**6.321.2.1** `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoMarshaller()` [inline]

**6.321.2.2** `virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` [inline, virtual]

## 6.321.3 Member Function Documentation

**6.321.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.321.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.321.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.321.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.321.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.321.3.6** virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.321.3.7** virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**

## 6.322 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1736).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.322.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1736).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.322.2 Constructor & Destructor Documentation

**6.322.2.1** `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoMarshaller()` [inline]

**6.322.2.2** `virtual activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` [inline, virtual]

## 6.322.3 Member Function Documentation

**6.322.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.322.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.322.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.322.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.322.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.322.3.6** virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.322.3.7** virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1740).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.323.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1740).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.323.2 Constructor & Destructor Documentation

**6.323.2.1** `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoMarshaller()` `[inline]`

**6.323.2.2** `virtual activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` `[inline, virtual]`

## 6.323.3 Member Function Documentation

**6.323.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.323.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.323.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.323.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.323.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.323.3.6** virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.323.3.7** virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h**

## 6.324 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1744).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.324.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1744).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.324.2 Constructor & Destructor Documentation

**6.324.2.1** `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoMarshaller()` [inline]

**6.324.2.2** `virtual activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` [inline, virtual]

## 6.324.3 Member Function Documentation

**6.324.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.324.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.324.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.324.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.324.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.324.3.6** virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.324.3.7** virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1748).

#include <src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.325.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1748).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.325.2 Constructor & Destructor Documentation

**6.325.2.1** `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::DestinationInfoMarshaller()` [inline]

**6.325.2.2** `virtual activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` [inline, virtual]

## 6.325.3 Member Function Documentation

**6.325.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.325.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.325.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.325.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.325.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.325.3.6** virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.325.3.7** virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DestinationInfoMarshaller.h**

## 6.326 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1752).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.326.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DestinationInfoMarshaller** (p.1752).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.326.2 Constructor & Destructor Documentation

**6.326.2.1** `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoMarshaller()` `[inline]`

**6.326.2.2** `virtual activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoMarshaller()` `[inline, virtual]`

## 6.326.3 Member Function Documentation

**6.326.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.326.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.326.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.326.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.326.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.326.3.6** virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.326.3.7** virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DestinationInfoMarshaller.h**

## 6.327 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

#include <src/main/activemq/cmsutil/DestinationResolver.h> Inheritance diagram for `activemq::cmsutil::DestinationResolver`:

### Public Member Functions

- virtual `~DestinationResolver()`
- virtual void `init(ResourceLifecycleManager *mgr)=0`  
*Initializes this destination resolver for use.*
- virtual void `destroy()`=0  
*Destroys any allocated resources.*
- virtual `cms::Destination * resolveDestinationName(cms::Session *session, const std::string &destName, bool pubSubDomain)=0` throw ( `cms::CMSException` )  
*Resolves the given name to a destination.*

### 6.327.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

### 6.327.2 Constructor & Destructor Documentation

- 6.327.2.1** virtual `activemq::cmsutil::DestinationResolver::~~DestinationResolver()` [inline, virtual]

### 6.327.3 Member Function Documentation

- 6.327.3.1** virtual void `activemq::cmsutil::DestinationResolver::destroy()` [pure virtual]

Destroys any allocated resources.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1822).

- 6.327.3.2** virtual void `activemq::cmsutil::DestinationResolver::init(ResourceLifecycleManager *mgr)` [pure virtual]

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1756)).

#### Parameters:

*mgr* the resource lifecycle manager.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1822).

**6.327.3.3** `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) throw ( cms::CMSException ) [pure virtual]`

Resolves the given name to a destination. If *pubSubDomain* is true, a topic will be returned, otherwise a queue will be returned.

**Parameters:**

*session* the session for which to retrieve resolve the destination.

*destName* the name to be resolved.

*pubSubDomain* If true, the name will be resolved to a Topic, otherwise a Queue.

**Returns:**

the resolved destination

**Exceptions:**

*cms::CMSException* (p. 1160) if resolution failed.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DestinationResolver.h`

## 6.328 activemq::commands::DiscoveryEvent Class Reference

#include <src/main/activemq/commands/DiscoveryEvent.h> Inheritance diagram for activemq::commands::DiscoveryEvent:

### Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DiscoveryEvent** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

### Static Public Attributes

- static const unsigned char **ID\_DISCOVERYEVENT** = 40

### Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

## 6.328.1 Constructor & Destructor Documentation

**6.328.1.1** `activemq::commands::DiscoveryEvent::DiscoveryEvent ()`

**6.328.1.2** `virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent ()`  
[virtual]

## 6.328.2 Member Function Documentation

**6.328.2.1** `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.328.2.2** `virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.328.2.3** `virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

- 6.328.2.4 `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ()`  
[virtual]
- 6.328.2.5 `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName ()`  
`const` [virtual]
- 6.328.2.6 `virtual unsigned char activemq::commands::DiscoveryEvent::getDataSetType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSet** (p. 1660) type copy.

Implements **activemq::commands::DataSet** (p. 1662).

- 6.328.2.7 `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ()`  
[virtual]
- 6.328.2.8 `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName ()`  
`const` [virtual]
- 6.328.2.9 `virtual void activemq::commands::DiscoveryEvent::setBrokerName`  
`(const std::string & brokerName)` [virtual]
- 6.328.2.10 `virtual void activemq::commands::DiscoveryEvent::setServiceName`  
`(const std::string & serviceName)` [virtual]
- 6.328.2.11 `virtual std::string activemq::commands::DiscoveryEvent::toString ()`  
`const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 833).



### 6.328.3 Field Documentation

- 6.328.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName`  
[protected]
- 6.328.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID_ -  
DISCOVERYEVENT = 40` [static]
- 6.328.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

## 6.329 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1762).

#include <src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.329.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1762).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.329.2 Constructor & Destructor Documentation

**6.329.2.1** `activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` `[inline]`

**6.329.2.2** `virtual activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` `[inline, virtual]`

## 6.329.3 Member Function Documentation

**6.329.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.329.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.329.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.329.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.329.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.329.3.6** virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.329.3.7** virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h

## 6.330 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1766).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.330.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1766).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.330.2 Constructor & Destructor Documentation

**6.330.2.1** `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` [inline]

**6.330.2.2** `virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` [inline, virtual]

## 6.330.3 Member Function Documentation

**6.330.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.330.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.330.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.330.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.330.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.330.3.6** virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.330.3.7** virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h

## 6.331 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1770).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.331.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1770).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.331.2 Constructor & Destructor Documentation

**6.331.2.1** `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` `[inline]`

**6.331.2.2** `virtual activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` `[inline, virtual]`

## 6.331.3 Member Function Documentation

**6.331.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.331.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.331.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.331.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.331.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.331.3.6** virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.331.3.7** virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h

## 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1774).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.332.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1774).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.332.2 Constructor & Destructor Documentation

**6.332.2.1** `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` `[inline]`

**6.332.2.2** `virtual activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` `[inline, virtual]`

## 6.332.3 Member Function Documentation

**6.332.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.332.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.332.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.332.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.332.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.332.3.6** virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.332.3.7** virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h

## 6.333 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1778).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.333.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1778).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.333.2 Constructor & Destructor Documentation

**6.333.2.1** `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` `[inline]`

**6.333.2.2** `virtual activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` `[inline, virtual]`

## 6.333.3 Member Function Documentation

**6.333.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.333.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.333.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.333.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.333.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.333.3.6** virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.333.3.7** virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h

## 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1782).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.334.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **DiscoveryEventMarshaller** (p.1782).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.334.2 Constructor & Destructor Documentation

**6.334.2.1** `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` `[inline]`

**6.334.2.2** `virtual activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` `[inline, virtual]`

## 6.334.3 Member Function Documentation

**6.334.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.334.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.334.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.334.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.334.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.334.3.6** virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.334.3.7** virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

## 6.335 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

### Public Member Functions

- **DispatchData** ()
- **DispatchData** (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)
- const decaf::lang::Pointer< commands::ConsumerId > & **getConsumerId** ()
- const decaf::lang::Pointer< commands::Message > & **getMessage** ()

### 6.335.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

### 6.335.2 Constructor & Destructor Documentation

**6.335.2.1** **activemq::core::DispatchData::DispatchData** () [inline]

**6.335.2.2** **activemq::core::DispatchData::DispatchData** (const decaf::lang::Pointer< commands::ConsumerId > & *consumer*, const decaf::lang::Pointer< commands::Message > & *message*) [inline]

### 6.335.3 Member Function Documentation

**6.335.3.1** const decaf::lang::Pointer<commands::ConsumerId>& **activemq::core::DispatchData::getConsumerId** () [inline]

**6.335.3.2** const decaf::lang::Pointer<commands::Message>& **activemq::core::DispatchData::getMessage** () [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**DispatchData.h**

## 6.336 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

#include <src/main/activemq/core/Dispatcher.h> Inheritance diagram for activemq::core::Dispatcher:

### Public Member Functions

- virtual **~Dispatcher** ()
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)=0  
*Dispatches a message to a particular consumer.*

#### 6.336.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

#### 6.336.2 Constructor & Destructor Documentation

**6.336.2.1** virtual **activemq::core::Dispatcher::~Dispatcher** () [inline, virtual]

#### 6.336.3 Member Function Documentation

**6.336.3.1** virtual void **activemq::core::Dispatcher::dispatch** (const **Pointer**< **MessageDispatch** > & *message*) [pure virtual]

Dispatches a message to a particular consumer.

#### Parameters:

*message* - the message to be dispatched.

Implemented in **activemq::core::ActiveMQConsumer** (p. 314), and **activemq::core::ActiveMQSession** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Dispatcher.h**

## 6.337 decaf::lang::Double Class Reference

#include <src/main/decaf/lang/Double.h> Inheritance diagram for decaf::lang::Double:

### Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Double** ()
- virtual int **compareTo** (const **Double** &d) const  
*Compares this **Double** (p. 1788) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Double** &d) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const double &d) const  
*Compares this **Double** (p. 1788) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const double &d) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const

*Answers the long value which the receiver represents.*

- bool **isInfinite** () const
- bool **isNaN** () const

## Static Public Member Functions

- static int **compare** (double d1, double d2)  
*Compares the two specified double values.*
- static long long **doubleToLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.*
- static long long **doubleToRawLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.*
- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)  
*Returns the double value corresponding to a given bit representation.*
- static double **parseDouble** (const std::string &value) throw (exceptions::NumberFormatException)  
*Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1788).*
- static std::string **toHexString** (double value)  
*Returns a hexadecimal string representation of the double argument.*
- static std::string **toString** (double value)  
*Returns a string representation of the double argument.*
- static **Double** **valueOf** (double value)  
*Returns a **Double** (p. 1788) instance representing the specified double value.*
- static **Double** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)  
*Returns a **Double** (p. 1788) instance that wraps a primitive double which is parsed from the string value passed.*

## Static Public Attributes

- static const int **SIZE** = 64  
*The size in bits of the primitive int type.*
- static const double **MAX\_VALUE**  
*The maximum value that the primitive type can hold.*

- static const double **MIN\_VALUE**  
*The minimum value that the primitive type can hold.*
- static const double **NaN**  
*Constant for the Not a **Number** (p. 2835) Value.*
- static const double **POSITIVE\_INFINITY**  
*Constant for Positive Infinity.*
- static const double **NEGATIVE\_INFINITY**  
*Constant for Negative Infinity.*

### 6.337.1 Constructor & Destructor Documentation

#### 6.337.1.1 `decaf::lang::Double::Double (double value)`

##### Parameters:

*value* - the primitive type to wrap

#### 6.337.1.2 `decaf::lang::Double::Double (const std::string & value) throw ( exceptions::NumberFormatException )`

##### Parameters:

*value* - the string to convert to a primitive type to wrap

#### 6.337.1.3 `virtual decaf::lang::Double::~~Double () [inline, virtual]`

### 6.337.2 Member Function Documentation

#### 6.337.2.1 `virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

##### Returns:

byte the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2835).

#### 6.337.2.2 `static int decaf::lang::Double::compare (double d1, double d2) [static]`

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: `new Double(d1).compareTo(new Double(d2))`

**Parameters:**

*d1* - the first double to compare

*d2* - the second double to compare

**Returns:**

the value 0 if d1 is numerically equal to d2; a value less than 0 if d1 is numerically less than d2; and a value greater than 0 if d1 is numerically greater than d2.

**6.337.2.3 virtual int decaf::lang::Double::compareTo (const double & d) const**  
[virtual]

Compares this **Double** (p. 1788) instance with another.

**Parameters:**

*d* - the **Double** (p. 1788) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< double > (p. 1214).

**6.337.2.4 virtual int decaf::lang::Double::compareTo (const Double & d) const**  
[virtual]

Compares this **Double** (p. 1788) instance with another.

**Parameters:**

*d* - the **Double** (p. 1788) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.337.2.5 static long long decaf::lang::Double::doubleToLongBits (double value)**  
[static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToLongBits` (except all NaN values are collapsed to a single "canonical" NaN value).

**Parameters:**

*value* - double to be converted

**Returns:**

the long long bits that make up the double

**6.337.2.6** `static long long decaf::lang::Double::doubleToRawLongBits (double value) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values. Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the `doubleToLongBits` method, `doubleToRawLongBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToRawLongBits`.

**Parameters:**

*value* - double to be converted

**Returns:**

the long long bits that make up the double

**6.337.2.7** `virtual double decaf::lang::Double::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements `decaf::lang::Number` (p. 2836).



**6.337.2.8** `bool decaf::lang::Double::equals (const double & d) const` [inline, virtual]

**Parameters:**

*d* - the **Double** (p. 1788) object to compare against.

**Returns:**

true if the two **Double** (p. 1788) Objects have the same value.

Implements **decaf::lang::Comparable< double >** (p. 1215).

**6.337.2.9** `bool decaf::lang::Double::equals (const Double & d) const` [inline]

**Parameters:**

*d* - the **Double** (p. 1788) object to compare against.

**Returns:**

true if the two **Double** (p. 1788) Objects have the same value.

**6.337.2.10** `virtual float decaf::lang::Double::floatValue () const` [inline, virtual]

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.337.2.11** `virtual int decaf::lang::Double::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.337.2.12** `static bool decaf::lang::Double::isInfinite (double value)` [static]

**Parameters:**

*value* - The double to check.

**Returns:**

true if the double is equal to infinity.

**6.337.2.13** `bool decaf::lang::Double::isInfinite () const`**Returns:**

true if the double is equal to positive infinity.

**6.337.2.14** `static bool decaf::lang::Double::isNaN (double value) [static]`**Parameters:**

*value* - The double to check.

**Returns:**

true if the double is equal to NaN.

**6.337.2.15** `bool decaf::lang::Double::isNaN () const`**Returns:**

true if the double is equal to NaN.

**6.337.2.16** `static double decaf::lang::Double::longBitsToDouble (long long bits) [static]`

Returns the double value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1792) method.

**Parameters:**

*bits* - the long long bits to convert to double

**Returns:**

the double converted from the bits

**6.337.2.17** `virtual long long decaf::lang::Double::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.337.2.18** `virtual bool decaf::lang::Double::operator< (const double & d) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*d* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1215).

**6.337.2.19** `virtual bool decaf::lang::Double::operator< (const Double & d) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*d* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.337.2.20** `virtual bool decaf::lang::Double::operator== (const double & d) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*d* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1215).

**6.337.2.21** `virtual bool decaf::lang::Double::operator== (const Double & d) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*d* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.337.2.22** `static double decaf::lang::Double::parseDouble (const std::string value)  
throw ( exceptions::NumberFormatException ) [static]`

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p.1788).

**Parameters:**

*value* - The string to parse to an double

**Returns:**

a double parsed from the passed string

**Exceptions:**

*NumberFormatException*

**6.337.2.23** `virtual short decaf::lang::Double::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns:**

short the value of the receiver.

Reimplemented from `decaf::lang::Number` (p.2837).

**6.337.2.24** `static std::string decaf::lang::Double::toHexString (double value)  
[static]`

Returns a hexadecimal string representation of the double argument. All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.2089) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

**Parameters:**

*value* - The double to convert to a string

**Returns:**

the Hex formatted double string.

**6.337.2.25 static std::string decaf::lang::Double::toString (double *value*) [static]**

Returns a string representation of the double argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude *m*:  
 o If *m* is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".  
 o If *m* is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".  
 o If *m* is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of *m*, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of *m*.  
 o If *m* is less than 10<sup>-3</sup> or greater than or equal to 10<sup>7</sup>, then it is represented in so-called "computerized scientific notation." Let *n* be the unique integer such that 10<sup>*n*</sup> ≤ *m* < 10<sup>*n*+1</sup>; then let *a* be the mathematically exact quotient of *m* and 10<sup>*n*</sup> so that 1 ≤ *a* < 10. The magnitude is then represented as the integer part of *a*, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of *a*, followed by the letter 'E', followed by a representation of *n* as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2089).

**Parameters:**

*value* - The double to convert to a string

**Returns:**

the formatted double string.

**6.337.2.26 std::string decaf::lang::Double::toString () const****Returns:**

this **Double** (p. 1788) Object as a **String** (p. 3665) Representation

**6.337.2.27 static Double decaf::lang::Double::valueOf (const std::string & *value*) throw ( exceptions::NumberFormatException ) [static]**

Returns a **Double** (p. 1788) instance that wraps a primitive double which is parsed from the string value passed.

**Parameters:**

*value* - the string to parse

**Returns:**

a new **Double** (p. 1788) instance wrapping the double parsed from value

**Exceptions:**

**NumberFormatException** on error.

**6.337.2.28**   `static Double decaf::lang::Double::valueOf (double value)`   [static]

Returns a **Double** (p. 1788) instance representing the specified double value.

**Parameters:**

*value* - double to wrap

**Returns:**

new **Double** (p. 1788) instance wrapping the primitive value

**6.337.3**   **Field Documentation****6.337.3.1**   `const double decaf::lang::Double::MAX_VALUE`   [static]

The maximum value that the primitive type can hold.

**6.337.3.2**   `const double decaf::lang::Double::MIN_VALUE`   [static]

The minimum value that the primitive type can hold.

**6.337.3.3**   `const double decaf::lang::Double::NaN`   [static]

Constant for the Not a **Number** (p. 2835) Value.

**6.337.3.4**   `const double decaf::lang::Double::NEGATIVE_INFINITY`   [static]

Constant for Negative Infinity.

**6.337.3.5**   `const double decaf::lang::Double::POSITIVE_INFINITY`   [static]

Constant for Positive Infinity.

**6.337.3.6**   `const int decaf::lang::Double::SIZE = 64`   [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Double.h`

## 6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h> Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

### Public Member Functions

- **DoubleArrayBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **DoubleArrayBuffer** (p. 1799) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **DoubleArrayBuffer** (double \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **DoubleArrayBuffer** (p. 1799) object that wraps the given array.*

- **DoubleArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*

- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)

*Create a **DoubleArrayBuffer** (p. 1799) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*

- virtual ~**DoubleArrayBuffer** ()
- virtual double \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the double array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928).*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset into the backing array where index zero starts.*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual **DoubleBuffer \* asReadOnlyBuffer** () const

*Creates a new, read-only double buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only double buffer which the caller then owns.*

- virtual **DoubleBuffer & compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns:**

*a reference to this **DoubleBuffer** (p. 1809).*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **DoubleBuffer \* duplicate** ()

*Creates a new double buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

*The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*a new double **Buffer** (p. 928) which the caller owns.*

- virtual double **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns:**

*the double at the current position.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return.



- virtual double **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the double is to be read.

**Returns:**

*the double that is located at the given index.*

**Exceptions:**

**IndexOutOfBoundsException** if index is not smaller than the buffer's limit

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible double array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.*

**Returns:**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns:**

*true if, and only if, this buffer is read-only.*

- virtual DoubleBuffer & **put** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the current position, and then increments the position.*

**Parameters:**

*value* The doubles value to be written.

**Returns:**

*a reference to this buffer.*

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual DoubleBuffer & **put** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the given index.*

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The doubles to write.

**Returns:**

*a reference to this buffer*

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or the index is negative.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual `DoubleBuffer * slice () const`

*Creates a new **DoubleBuffer** (p. 1809) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the newly create **DoubleBuffer** (p. 1809) which the caller owns.*

## Protected Member Functions

- virtual void `setReadOnly (bool value)`

*Sets this **DoubleArrayBuffer** (p. 1799) as Read-Only or not Read-Only.*

### 6.338.1 Constructor & Destructor Documentation

#### 6.338.1.1 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (int capacity, bool readOnly = false) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **DoubleArrayBuffer** (p. 1799) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

##### Parameters:

*size* The size of the array, this is the limit we read and write to.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

##### Exceptions:

***IllegalArgumentException*** if the capacity value is negative.

#### 6.338.1.2 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * array, int size, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **DoubleArrayBuffer** (p. 1799) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

##### Parameters:

*array* The actual array to wrap.

*size* The size of the given array.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

#### 6.338.1.3 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset. The capacity and limit of the new **DoubleArrayBuffer** (p.1799) will be that of the remaining capacity of the passed buffer.

#### Parameters:

*array* The ByteArrayAdapter to wrap.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if array is NULL

*IndexOutOfBoundsException* if offset + length is greater than array size.

#### 6.338.1.4 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & other)

Create a **DoubleArrayBuffer** (p.1799) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters:

*other* The **DoubleArrayBuffer** (p.1799) this one is to mirror.

#### 6.338.1.5 virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer () [virtual]

### 6.338.2 Member Function Documentation

#### 6.338.2.1 virtual double\* decaf::internal::nio::DoubleArrayBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1812).

**6.338.2.2** `virtual int decaf::internal::nio::DoubleArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1812).

**6.338.2.3** `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer ()  
const [virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1813).

**6.338.2.4 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact  
( ) throw ( decaf::nio::ReadOnlyBufferException ) [virtual]**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **DoubleBuffer** (p. 1809).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1813).

**6.338.2.5 virtual DoubleBuffer\* decaf::internal::nio::DoubleArrayBuffer::duplicate  
( ) [virtual]**

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new double **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1813).

**6.338.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get (int *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]**

Absolute get method.

Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the double is to be read.

**Returns:**

the double that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 1815).

**6.338.2.7** virtual double decaf::internal::nio::DoubleArrayBuffer::get () throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the double at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implements **decaf::nio::DoubleBuffer** (p. 1815).

**6.338.2.8** virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1815).

**6.338.2.9** virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 931).

**6.338.2.10** virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (int *index*, double *value*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given doubles into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The doubles to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or the index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1816).

**6.338.2.11** virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (double *value*) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The doubles value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1816).

**6.338.2.12** virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 1799) as Read-Only or not Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.338.2.13** virtual DoubleBuffer\* decaf::internal::nio::DoubleArrayBuffer::slice () const [virtual]

Creates a new **DoubleBuffer** (p. 1809) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **DoubleBuffer** (p. 1809) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1818).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`



## 6.339 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

#include <src/main/decaf/nio/DoubleBuffer.h> Inheritance diagram for decaf::nio::DoubleBuffer:

### Public Member Functions

- virtual **~DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the double array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **DoubleBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only double buffer that shares this buffer's content.*
- virtual **DoubleBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **DoubleBuffer** \* **duplicate** ()=0  
*Creates a new double buffer that shares this buffer's content.*
- virtual double **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual double **get** (int index) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **DoubleBuffer** & **get** (std::vector< double > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **DoubleBuffer** & **get** (double \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible double array.*
- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the doubles remaining in the given source buffer into this buffer.*

- **DoubleBuffer** & **put** (const double \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers doubles into this buffer from the given source array.*

- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source doubles array into this buffer.*

- virtual **DoubleBuffer** & **put** (double value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given doubles into this buffer at the current position, and then increments the position.*

- virtual **DoubleBuffer** & **put** (int index, double value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given doubles into this buffer at the given index.*

- virtual **DoubleBuffer** \* **slice** () const =0

*Creates a new **DoubleBuffer** (p. 1809) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **DoubleBuffer** &value) const

- virtual bool **equals** (const **DoubleBuffer** &value) const

- virtual bool **operator==** (const **DoubleBuffer** &value) const

- virtual bool **operator<** (const **DoubleBuffer** &value) const

## Static Public Member Functions

- static **DoubleBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new **DoubleBuffer** (p. 1809).*

- static **DoubleBuffer** \* **wrap** (double \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **DoubleBuffer** (p. 1809).*

- static **DoubleBuffer** \* **wrap** (std::vector< double > &buffer)

*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1809).*

## Protected Member Functions

- **DoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **DoubleBuffer** (p. 1809) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.339.1 Detailed Description

This class defines four categories of operations upon double buffers:
 

- o Absolute and relative get and put methods that read and write single doubles;
- o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer

 o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

Since:

1.0

### 6.339.2 Constructor & Destructor Documentation

#### 6.339.2.1 decaf::nio::DoubleBuffer::DoubleBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **DoubleBuffer** (p. 1809) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

*capacity* The size and limit of the **Buffer** (p. 928) in doubles

Exceptions:

*IllegalArgumentException* if capacity is negative.

#### 6.339.2.2 virtual decaf::nio::DoubleBuffer::~~DoubleBuffer () [inline, virtual]

### 6.339.3 Member Function Documentation

#### 6.339.3.1 static DoubleBuffer\* decaf::nio::DoubleBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new **DoubleBuffer** (p. 1809). The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters:**

*capacity* The size of the Double buffer in doubles.

**Returns:**

the **DoubleBuffer** (p. 1809) that was allocated, caller owns.

**Exceptions:**

*IllegalArgumentException* is the capacity value is negative.

**6.339.3.2** `virtual double* decaf::nio::DoubleBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the double array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1803).

**6.339.3.3** `virtual int decaf::nio::DoubleBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1804).

**6.339.3.4 virtual DoubleBuffer\* decaf::nio::DoubleBuffer::asReadOnlyBuffer ()**  
**const** [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1804).

**6.339.3.5 virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact () throw (**  
**ReadOnlyBufferException )** [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **DoubleBuffer** (p. 1809).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1805).

**6.339.3.6 virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer &**  
**value) const** [virtual]**6.339.3.7 virtual DoubleBuffer\* decaf::nio::DoubleBuffer::duplicate ()** [pure  
virtual]

Creates a new double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new double **Buffer** (p. 928) which the caller owns.

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1805).

**6.339.3.8** `virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & value) const` [virtual]

**6.339.3.9** `DoubleBuffer& decaf::nio::DoubleBuffer::get (double * buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method. This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies `length` doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

#### Parameters:

*buffer* The pointer to an allocated buffer to fill.  
*size* The size of the buffer passed.  
*offset* The position in the buffer to start filling.  
*length* The amount of data to put in the passed buffer.

#### Returns:

a reference to this **Buffer** (p. 928).

#### Exceptions:

**BufferUnderflowException** (p. 957) if there are fewer than `length` doubles remaining in this buffer  
**NullPointerException** if the passed buffer is null.  
**IndexOutOfBoundsException** if the preconditions of `size`, `offset`, or `length` are not met.

**6.339.3.10** `DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > buffer) throw ( BufferUnderflowException )`

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

#### Returns:

a reference to this **Buffer** (p. 928).

#### Exceptions:

**BufferUnderflowException** (p. 957) if there are fewer than `length` doubles remaining in this buffer

**6.339.3.11 virtual double decaf::nio::DoubleBuffer::get (int *index*) const throw (lang::exceptions::IndexOutOfBoundsException ) [pure virtual]**

Absolute get method. Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the double is to be read.

**Returns:**

the double that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1805).

**6.339.3.12 virtual double decaf::nio::DoubleBuffer::get () throw (BufferUnderflowException ) [pure virtual]**

Relative get method. Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the double at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1806).

**6.339.3.13 virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]**

Tells whether or not this buffer is backed by an accessible double array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1806).

- 6.339.3.14** `virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & value) const` [virtual]
- 6.339.3.15** `virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & value) const` [virtual]
- 6.339.3.16** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes the given doubles into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The doubles to write.

**Returns:**

a reference to this buffer

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or the index is negative.  
*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1806).

- 6.339.3.17** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The doubles value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.  
*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1807).

- 6.339.3.18** `DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & buffer) throw ( BufferOverflowException, ReadOnlyBufferException )`

This method transfers the entire content of the given source doubles array into this buffer. This is the same as calling `put ( &buffer[0], 0, buffer.size() )`.



**Parameters:**

*buffer* The buffer whose contents are copied to this **DoubleBuffer** (p. 1809).

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**6.339.3.19 DoubleBuffer& decaf::nio::DoubleBuffer::put (const double \* *buffer*, int *size*, int *offset*, int *length*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers doubles into this buffer from the given source array. If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no doubles are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters:**

*buffer* The array from which doubles are to be read.

*size* The size of the buffer passed.

*offset* The offset within the array of the first char to be read.

*length* The number of doubles to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

**NullPointerException** if the passed buffer is null.

**IndexOutOfBoundsException** if the preconditions of `size`, `offset`, or `length` are not met.

**6.339.3.20 DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & *src*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )**

This method transfers the doubles remaining in the given source buffer into this buffer. If there are more doubles remaining in the source buffer than in this buffer, that is, if `src.remaining() >`

**remaining()** (p. 933), then no doubles are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies  $n = \text{src.remaining}()$  doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by  $n$ .

**Parameters:**

*src* The buffer to take doubles from an place in this one.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer for the remaining doubles in the source buffer

**IllegalArgumentException** if the source buffer is this buffer.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

### 6.339.3.21 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const` [pure virtual]

Creates a new **DoubleBuffer** (p. 1809) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **DoubleBuffer** (p. 1809) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1807).

### 6.339.3.22 `virtual std::string decaf::nio::DoubleBuffer::toString () const` [virtual]

**Returns:**

a `std::string` describing this object

### 6.339.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector< double > & buffer)` [static]

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1809). The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new **DoubleBuffer** (p.1809) that is backed by *buffer*, caller owns.

**6.339.3.24** `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap  
(double * array, int size, int offset, int length)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **DoubleBuffer** (p.1809). The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*array* The array that will back the new buffer.

*size* The size of the passed in array.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new **DoubleBuffer** (p.1809) that is backed by *buffer*, caller owns.

**Exceptions:**

*NullPointerException* if the array pointer is NULL.

*IndexOutOfBoundsException* if the preconditions of *size*, *offset*, or *length* are not met.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/DoubleBuffer.h`

## 6.340 decaf::lang::DYNAMIC\_CAST\_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.341 activemq::cmsutil::DynamicDestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

`#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>` Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

### Data Structures

- class `SessionResolver`

*Manages maps of names to topics and queues for a single session.*

### Public Member Functions

- `DynamicDestinationResolver ()`
- `virtual ~DynamicDestinationResolver ()`
- `virtual void init (ResourceLifecycleManager *mgr)`

*Initializes this destination resolver for use.*

- `virtual void destroy ()`

*Destroys any allocated resources.*

- `virtual cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain) throw ( cms::CMSEException )`

*Resolves the given name to a destination.*

### Protected Member Functions

- `DynamicDestinationResolver (const DynamicDestinationResolver &)`
- `DynamicDestinationResolver & operator= (const DynamicDestinationResolver &)`

#### 6.341.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

## 6.341.2 Constructor & Destructor Documentation

- 6.341.2.1** `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver (const DynamicDestinationResolver &) [inline, protected]`
- 6.341.2.2** `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ()`
- 6.341.2.3** `virtual  
activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver  
() [virtual]`

## 6.341.3 Member Function Documentation

- 6.341.3.1** `virtual void activemq::cmsutil::DynamicDestinationResolver::destroy () [virtual]`

Destroys any allocated resources.

Implements `activemq::cmsutil::DestinationResolver` (p. 1756).

- 6.341.3.2** `virtual void activemq::cmsutil::DynamicDestinationResolver::init (ResourceLifecycleManager * mgr) [inline, virtual]`

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1822)).

### Parameters:

*mgr* the resource lifecycle manager.

Implements `activemq::cmsutil::DestinationResolver` (p. 1756).

- 6.341.3.3** `DynamicDestinationResolver& activemq::cmsutil::DynamicDestinationResolver::operator= (const DynamicDestinationResolver &) [inline, protected]`

- 6.341.3.4** `virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) throw ( cms::CMSException ) [virtual]`

Resolves the given name to a destination. If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

### Parameters:

*session* the session for which to retrieve resolve the destination.

*destName* the name to be resolved.

*pubSubDomain* If true, the name will be resolved to a Topic, otherwise a Queue.

### Returns:

the resolved destination

**Exceptions:**

*cms::CMSException* (p. 1160) if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p. 1757).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DynamicDestinationResolver.h`

## 6.342 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

### Public Member Functions

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>  
class decaf::util::Map< K, V, COMPARATOR >::Entry
```

### 6.342.1 Constructor & Destructor Documentation

**6.342.1.1** `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry () [inline]`

**6.342.1.2** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry () [inline, virtual]`

### 6.342.2 Member Function Documentation

**6.342.2.1** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey () const [pure virtual]`

**6.342.2.2** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue () const [pure virtual]`

**6.342.2.3** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue (const V & value) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`



## 6.343 decaf::io::EOFException Class Reference

#include <src/main/decaf/io/EOFException.h> Inheritance diagram for decaf::io::EOFException:

### Public Member Functions

- **EOFException** () throw ()  
*Default Constructor.*
- **EOFException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **EOFException** (const EOFException &ex) throw ()  
*Copy Constructor.*
- **EOFException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **EOFException** (const std::exception \*cause) throw ()  
*Constructor.*
- **EOFException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **EOFException** \* clone () const  
*Clones this exception.*
- virtual ~**EOFException** () throw ()

### 6.343.1 Constructor & Destructor Documentation

#### 6.343.1.1 decaf::io::EOFException::EOFException () throw () [inline]

Default Constructor.

#### 6.343.1.2 decaf::io::EOFException::EOFException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

### 6.343.1.3 decaf::io::EOFException::EOFException (const EOFException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy, which is an instance of this type

### 6.343.1.4 decaf::io::EOFException::EOFException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.343.1.5 decaf::io::EOFException::EOFException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.343.1.6 decaf::io::EOFException::EOFException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor.

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.343.1.7** `virtual decaf::io::EOFException::~~EOFException () throw () [inline, virtual]`

## 6.343.2 Member Function Documentation

**6.343.2.1** `virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

## 6.344 decaf::util::logging::ErrorManager Class Reference

**ErrorManager** (p. 1828) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1978) during Logging.

```
#include <src/main/decaf/util/logging/ErrorManager.h>
```

### Public Member Functions

- **ErrorManager** ()
- virtual **~ErrorMessage** ()
- virtual void **error** (const std::string &message, **decaf::lang::Exception** \*ex, int code)

*The error method is called when a **Handler** (p. 1978) failure occurs.*

### Static Public Attributes

- static const int **GENERIC\_FAILURE**  
*GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.*
- static const int **WRITE\_FAILURE**  
*WRITE\_FAILURE is used when a write to an output stream fails.*
- static const int **FLUSH\_FAILURE**  
*FLUSH\_FAILURE is used when a flush to an output stream fails.*
- static const int **CLOSE\_FAILURE**  
*CLOSE\_FAILURE is used when a close of an output stream fails.*
- static const int **OPEN\_FAILURE**  
*OPEN\_FAILURE is used when an open of an output stream fails.*
- static const int **FORMAT\_FAILURE**  
*FORMAT\_FAILURE is used when formatting fails for any reason.*

### 6.344.1 Detailed Description

**ErrorManager** (p. 1828) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1978) during Logging. When processing **logging** (p. 175) output, if a **Handler** (p. 1978) encounters problems then rather than throwing an Exception back to the issuer of the **logging** (p. 175) call (who is unlikely to be interested) the **Handler** (p. 1978) should call its associated **ErrorManager** (p. 1828).

Since:

1.0

## 6.344.2 Constructor & Destructor Documentation

**6.344.2.1** `decaf::util::logging::ErrorManager::ErrorManager ()`

**6.344.2.2** `virtual decaf::util::logging::ErrorManager::~~ErrorManager ()` [virtual]

## 6.344.3 Member Function Documentation

**6.344.3.1** `virtual void decaf::util::logging::ErrorManager::error (const std::string & message, decaf::lang::Exception * ex, int code)` [virtual]

The error method is called when a **Handler** (p.1978) failure occurs. This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

### Parameters:

*msg* - a descriptive string (may be empty)

*ex* - an exception (may be NULL)

*code* (p. 1183) - an error **code** (p.1183) defined in **ErrorManager** (p.1828)

## 6.344.4 Field Documentation

**6.344.4.1** `const int decaf::util::logging::ErrorManager::CLOSE_FAILURE`  
[static]

CLOSE\_FAILURE is used when a close of an output stream fails.

**6.344.4.2** `const int decaf::util::logging::ErrorManager::FLUSH_FAILURE`  
[static]

FLUSH\_FAILURE is used when a flush to an output stream fails.

**6.344.4.3** `const int decaf::util::logging::ErrorManager::FORMAT_FAILURE`  
[static]

FORMAT\_FAILURE is used when formatting fails for any reason.

**6.344.4.4** `const int decaf::util::logging::ErrorManager::GENERIC_FAILURE`  
[static]

GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.

**6.344.4.5** `const int decaf::util::logging::ErrorManager::OPEN_FAILURE` [static]

OPEN\_FAILURE is used when an open of an output stream fails.

#### 6.344.4.6 `const int decaf::util::logging::ErrorManager::WRITE_FAILURE` [static]

WRITE\_FAILURE is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ErrorManager.h`

## 6.345 decaf::lang::Exception Class Reference

#include <src/main/decaf/lang/Exception.h> Inheritance diagram for decaf::lang::Exception:

### Public Member Functions

- **Exception** () throw ()  
*Default Constructor.*
- **Exception** (const **Exception** &ex) throw ()  
*Copy Constructor.*
- **Exception** (const std::exception \***cause**) throw ()  
*Constructor.*
- **Exception** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **Exception** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const  
*Gets the message for this exception.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 159) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \***cause**)  
*Initializes the contained cause exception with the one given.*
- virtual const char \* **what** () const throw ()  
*Implement method from std::exception.*
- virtual void **setMessage** (const char \*msg,...)  
*Sets the cause for this exception.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual **Exception** \* **clone** () const  
*Clones this exception.*

- virtual `std::vector< std::pair< std::string, int > > getStackTrace () const`  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void `printStackTrace () const`  
*Prints the stack trace to `std::err`.*
- virtual void `printStackTrace (std::ostream &stream) const`  
*Prints the stack trace to the given output stream.*
- virtual `std::string getStackTraceString () const`  
*Gets the stack trace as one contiguous string.*
- `Exception & operator= (const Exception &ex)`  
*Assignment operator.*

## Protected Member Functions

- virtual void `setStackTrace (const std::vector< std::pair< std::string, int > > &trace)`
- virtual void `buildMessage (const char *format, va_list &args)`

## Protected Attributes

- `std::string message`  
*The cause of this exception.*
- `std::exception * cause`  
*The **Exception** (p. 1831) that caused this one to be thrown.*
- `std::vector< std::pair< std::string, int > > stackTrace`  
*The stack trace.*

## 6.345.1 Constructor & Destructor Documentation

### 6.345.1.1 `decaf::lang::Exception::Exception () throw ()`

Default Constructor.

Referenced by `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException()`,  
`decaf::util::concurrent::CancellationException::CancellationException()`,  
`decaf::util::concurrent::ExecutionException::ExecutionException()`,  
`decaf::net::HttpRetryException::HttpRetryException()`, `decaf::net::MalformedURLException::MalformedURLException()`,  
`decaf::net::ProtocolException::ProtocolException()`, `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException()`,  
`decaf::net::SocketTimeoutException::SocketTimeoutException()`,  
`decaf::util::concurrent::TimeoutException::TimeoutException()`,  
`decaf::net::UnknownHostException::UnknownHostException()`, and `decaf::net::UnknownServiceException::UnknownServiceException()`.



**6.345.1.2 decaf::lang::Exception::Exception (const Exception & *ex*) throw ()**

Copy Constructor.

**Parameters:**

*ex* The Exception (p.1831) instance to copy.

**6.345.1.3 decaf::lang::Exception::Exception (const std::exception \* *cause*) throw ()**

Constructor.

**Parameters:**

*cause* **Pointer** (p.2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.345.1.4 decaf::lang::Exception::Exception (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.345.1.5 decaf::lang::Exception::Exception (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw ()**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.345.1.6** virtual `decaf::lang::Exception::~~Exception () throw ()` [virtual]

## 6.345.2 Member Function Documentation

**6.345.2.1** virtual void `decaf::lang::Exception::buildMessage (const char * format, va_list & vargs)` [protected, virtual]

Referenced by `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

**6.345.2.2** virtual `Exception* decaf::lang::Exception::clone () const` [virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

Copy of this **Exception** (p.1831) object

Implements **decaf::lang::Throwable** (p. 3784).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 358), **activemq::exceptions::BrokerException** (p. 867), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2873), **decaf::io::EOFException** (p. 1827), **decaf::io::InterruptedIOException** (p. 2129), **decaf::io::IOException** (p. 2144), **decaf::io::UnsupportedEncodingException** (p. 3915), **decaf::io::UTFDataFormatException** (p. 3968), **decaf::lang::exceptions::ClassCastException** (p. 1147), **decaf::lang::exceptions::IllegalArgumentException** (p. 1993), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1996), **decaf::lang::exceptions::IllegalStateException** (p. 2000), **decaf::lang::exceptions::IllegalThreadStateException** (p. 2003), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 2010), **decaf::lang::exceptions::InterruptedException** (p. 2126), **decaf::lang::exceptions::InvalidStateException** (p. 2141), **decaf::lang::exceptions::NoSuchElementException** (p. 2828), **decaf::lang::exceptions::NullPointerException** (p. 2834), **decaf::lang::exceptions::NumberFormatException** (p. 2840), **decaf::lang::exceptions::RuntimeException** (p. 3330), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3918), **decaf::net::BindException** (p. 838), **decaf::net::ConnectException** (p. 1261), **decaf::net::HttpRetryException** (p. 1988), **decaf::net::MalformedURLException** (p. 2458), **decaf::net::NoRouteToHostException** (p. 2822), **decaf::net::PortUnreachableException** (p. 2975), **decaf::net::ProtocolException** (p. 3139), **decaf::net::SocketException** (p. 3522), **decaf::net::SocketTimeoutException** (p. 3544), **decaf::net::UnknownHostException** (p. 3909), **decaf::net::UnknownServiceException** (p. 3912), **decaf::net::URISyntaxException** (p. 3950), **decaf::nio::BufferOverflowException** (p. 956), **decaf::nio::BufferUnderflowException** (p. 959), **decaf::nio::InvalidMarkException** (p. 2137), **decaf::nio::ReadOnlyBufferException** (p. 3171), **decaf::security::cert::CertificateEncodingException** (p. 1091), **decaf::security::cert::CertificateException** (p. 1093), **decaf::security::cert::CertificateExpiredException** (p. 1095), **decaf::security::cert::CertificateNotYetValidException** (p. 1097), **decaf::security::cert::CertificateParsingException** (p. 1099), **decaf::security::GeneralSecurityException** (p. 1973), **de-**

**caf::security::InvalidKeyException** (p. 2134), **decaf::security::KeyException** (p. 2297),  
**decaf::security::KeyManagementException** (p. 2300),  
**decaf::security::NoSuchAlgorithmException** (p. 2825), **decaf::security::NoSuchProviderException** (p. 2831),  
**decaf::security::SignatureException** (p. 3499), **decaf::util::concurrent::BrokenBarrierException** (p. 862),  
**decaf::util::concurrent::CancellationException** (p. 1085), **decaf::util::concurrent::ExecutionException** (p. 1868),  
**decaf::util::concurrent::RejectedExecutionException** (p. 3191),  
**decaf::util::concurrent::TimeoutException** (p. 3789), **decaf::util::zip::DataFormatException** (p. 1557), and **decaf::util::zip::ZipException** (p. 4066).

#### 6.345.2.3 virtual const std::exception\* decaf::lang::Exception::getCause () const [inline, virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 159) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

##### Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 3785).

#### 6.345.2.4 virtual std::string decaf::lang::Exception::getMessage () const [inline, virtual]

Gets the message for this exception.

##### Returns:

Text formatted error message

Implements **decaf::lang::Throwable** (p. 3785).

#### 6.345.2.5 virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace () const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown. The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

##### Returns:

the stack trace.

Implements **decaf::lang::Throwable** (p. 3785).

**6.345.2.6** `virtual std::string decaf::lang::Exception::getStackTraceString () const [virtual]`

Gets the stack trace as one contiguous string.

**Returns:**

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 3786).

**6.345.2.7** `virtual void decaf::lang::Exception::initCause (const std::exception * cause) [virtual]`

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

**Parameters:**

*cause* The exception that was the cause of this one.

Implements **decaf::lang::Throwable** (p. 3786).

**6.345.2.8** `Exception& decaf::lang::Exception::operator= (const Exception & ex)`

Assignment operator.

**Parameters:**

*ex* const reference to another **Exception** (p. 1831)

**6.345.2.9** `virtual void decaf::lang::Exception::printStackTrace (std::ostream & stream) const [virtual]`

Prints the stack trace to the given output stream.

**Parameters:**

*stream* the target output stream.

Implements **decaf::lang::Throwable** (p. 3786).

**6.345.2.10** `virtual void decaf::lang::Exception::printStackTrace () const [virtual]`

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 3786).

**6.345.2.11** `virtual void decaf::lang::Exception::setMark (const char * file, const int lineNumber) [virtual]`

Adds a file/line number to the stack trace.

**Parameters:**

*file* The name of the file calling this method (use `__FILE__`).

*lineNumber* The line number in the calling file (use `__LINE__`).

Implements **decaf::lang::Throwable** (p. 3786).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`, and `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

### 6.345.2.12 **virtual void decaf::lang::Exception::setMessage (const char \* *msg*, ...)** [virtual]

Sets the cause for this exception.

**Parameters:**

*msg* the format string for the msg.

... params to format into the string

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

### 6.345.2.13 **virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & *trace*)** [protected, virtual]

### 6.345.2.14 **virtual const char\* decaf::lang::Exception::what () const throw ()** [inline, virtual]

Implement method from `std::exception`.

**Returns:**

the const char\* of `getMessage()` (p. 1835).

## 6.345.3 Field Documentation

### 6.345.3.1 **std::exception\* decaf::lang::Exception::cause** [protected]

The **Exception** (p. 1831) that caused this one to be thrown.

### 6.345.3.2 **std::string decaf::lang::Exception::message** [protected]

The cause of this exception.

### 6.345.3.3 **std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace** [protected]

The stack trace.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Exception.h`

## 6.346 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1838) that is registered with the `Connection` (p. 1262).

```
#include <src/main/cms/ExceptionListener.h>
```

### Public Member Functions

- virtual `~ExceptionListener ()`
- virtual void `onException (const cms::CMSException &ex)=0`  
*Called when an exception occurs.*

### 6.346.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1838) that is registered with the `Connection` (p. 1262). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since:

1.0

### 6.346.2 Constructor & Destructor Documentation

**6.346.2.1** virtual `cms::ExceptionListener::~ExceptionListener ()` [inline, virtual]

### 6.346.3 Member Function Documentation

**6.346.3.1** virtual void `cms::ExceptionListener::onException (const cms::CMSException & ex)` [pure virtual]

Called when an exception occurs. Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters:

*ex* Exception Object that occurred.

The documentation for this class was generated from the following file:

- `src/main/cms/ExceptionListener.h`

## 6.347 activemq::commands::ExceptionResponse Class Reference

#include <src/main/activemq/commands/ExceptionResponse.h> Inheritance diagram for activemq::commands::ExceptionResponse:

### Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ExceptionResponse \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

### Static Public Attributes

- static const unsigned char **ID\_EXCEPTIONRESPONSE** = 31

### Protected Attributes

- **Pointer**< **BrokerError** > **exception**

### 6.347.1 Constructor & Destructor Documentation

**6.347.1.1** `activemq::commands::ExceptionResponse::ExceptionResponse ()`

**6.347.1.2** `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse () [virtual]`

### 6.347.2 Member Function Documentation

**6.347.2.1** `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.347.2.2** `virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.347.2.3** `virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.347.2.4** `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.



**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from **activemq::commands::Response** (p. 3287).

**6.347.2.5** `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ()`  
[virtual]

**6.347.2.6** `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ()`  
const [virtual]

**6.347.2.7** `virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & exception)` [virtual]

**6.347.2.8** `virtual std::string activemq::commands::ExceptionResponse::toString ()`  
const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3287).

**6.347.3 Field Documentation**

**6.347.3.1** `Pointer<BrokerError> activemq::commands::ExceptionResponse::exception`  
[protected]

**6.347.3.2** `const unsigned char activemq::commands::ExceptionResponse::ID_ - EXCEPTIONRESPONSE = 31` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ExceptionResponse.h`

## 6.348 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1842).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.348.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1842).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.348.2 Constructor & Destructor Documentation

**6.348.2.1** `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.348.2.2** `virtual  
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.348.3 Member Function Documentation

**6.348.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`  
(p. 3321).

**6.348.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`  
(p. 3321).

**6.348.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.348.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.348.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.348.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3323).

**6.348.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3324).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h

## 6.349 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1846).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.349.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1846).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.349.2 Constructor & Destructor Documentation

**6.349.2.1** `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.349.2.2** `virtual  
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.349.3 Member Function Documentation

**6.349.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller`  
(p. 3301).

**6.349.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller`  
(p. 3301).

**6.349.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.349.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.349.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).



**6.349.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3303).

**6.349.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3304).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h

## 6.350 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1850).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.350.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1850).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.350.2 Constructor & Destructor Documentation

**6.350.2.1** `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.350.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.350.3 Member Function Documentation

**6.350.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`  
(p. 3311).

**6.350.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`  
(p. 3311).

**6.350.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.350.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.350.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.350.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3313).

**6.350.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3314).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h

## 6.351 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1854).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.351.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1854).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.351.2 Constructor & Destructor Documentation

**6.351.2.1** `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.351.2.2** `virtual  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.351.3 Member Function Documentation

**6.351.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`  
(p. 3306).

**6.351.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`  
(p. 3306).

**6.351.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.351.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.351.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).



**6.351.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3308).

**6.351.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3309).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h

## 6.352 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1858).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.352.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1858).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.352.2 Constructor & Destructor Documentation

**6.352.2.1** `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.352.2.2** `virtual  
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.352.3 Member Function Documentation

**6.352.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller`  
(p. 3296).

**6.352.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller`  
(p. 3296).

**6.352.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.352.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.352.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.352.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3298).

**6.352.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h

## 6.353 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1862).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.353.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1862).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.353.2 Constructor & Destructor Documentation

**6.353.2.1** `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionRe  
( ) [inline]`

**6.353.2.2** `virtual  
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~~ExceptionR  
( ) [inline, virtual]`

## 6.353.3 Member Function Documentation

**6.353.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`  
(p. 3316).

**6.353.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`  
(p. 3316).

**6.353.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.353.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).

**6.353.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).



**6.353.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3318).

**6.353.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h

## 6.354 decaf::util::concurrent::ExecutionException Class Reference

#include <src/main/decaf/util/concurrent/ExecutionException.h> Inheritance diagram for decaf::util::concurrent::ExecutionException:

### Public Member Functions

- **ExecutionException** () throw ()  
*Default Constructor.*
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ExecutionException** (const **ExecutionException** &ex) throw ()  
*Copy Constructor.*
- **ExecutionException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ExecutionException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ExecutionException** \* clone () const  
*Clones this exception.*
- virtual ~**ExecutionException** () throw ()

### 6.354.1 Constructor & Destructor Documentation

#### 6.354.1.1 decaf::util::concurrent::ExecutionException::ExecutionException () throw () [inline]

Default Constructor.

#### 6.354.1.2 decaf::util::concurrent::ExecutionException::ExecutionException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** - An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

#### 6.354.1.3 decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & *ex*) throw () [inline]

Copy Constructor.

##### Parameters:

*ex* - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

#### 6.354.1.4 decaf::util::concurrent::ExecutionException::ExecutionException (const std::exception \* *cause*) throw () [inline]

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.354.1.5 decaf::util::concurrent::ExecutionException::ExecutionException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*msg* - The message to report

... - The list of primitives that are formatted into the message

#### 6.354.1.6 decaf::util::concurrent::ExecutionException::ExecutionException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*cause* - The exception that was the cause for this one to be thrown.

*msg* - The message to report

... - list of primitives that are formatted into the message

**6.354.1.7** `virtual decaf::util::concurrent::ExecutionException::~~ExecutionException  
() throw () [inline, virtual]`

## **6.354.2 Member Function Documentation**

**6.354.2.1** `virtual ExecutionException* de-  
caf::util::concurrent::ExecutionException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutionException.h`

## 6.355 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 3325) tasks.

#include <src/main/decaf/util/concurrent/Executor.h> Inheritance diagram for decaf::util::concurrent::Executor:

### Public Member Functions

- virtual **~Executor** ()
- virtual void **execute** (Runnable \*command)=0 throw ( decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException )

*Executes the given command at some time in the future.*

### 6.355.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 3325) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1869) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()).start()` for each of a set of tasks, you might use:

```
Executor (p.1869) executor = anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p. 1869) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p.1869) {
public:

    void execute( Runnable* r ) (p.1870) {
        r->run();
    }

}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p.1869) {
public:
    std::vector<Thread*gt; threads;
```

```

void execute( Runnable* r ) (p.1870) {
    threads.push_back( new Thread( r ) );
    threads.rbegin()->start();
}

}

```

The `Executor` (p.1869) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p.1871), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p.??) class provides an extensible thread pool implementation. The **decaf.util.concurrent.Executor** (p.??) class provides convenient factory methods for these Executors.

**Since:**

1.0

## 6.355.2 Constructor & Destructor Documentation

**6.355.2.1** `virtual decaf::util::concurrent::Executor::~~Executor ()` [inline, virtual]

## 6.355.3 Member Function Documentation

**6.355.3.1** `virtual void decaf::util::concurrent::Executor::execute (Runnable * command) throw ( decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException )` [pure virtual]

Executes the given command at some time in the future. The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the `Executor` (p.1869) implementation.

**Parameters:**

*command* the runnable task

**Exceptions:**

***RejectedExecutionException*** (p. 3189) if this task cannot be accepted for execution.

***NullPointerException*** if command is null

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Executor.h`

## 6.356 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p.1869) that provides methods to manage termination and methods that can produce a **Future** (p.1966) for tracking progress of one or more asynchronous tasks.

#include <src/main/decaf/util/concurrent/ExecutorService.h> Inheritance diagram for decaf::util::concurrent::ExecutorService:

### Public Member Functions

- virtual  $\sim$ **ExecutorService** ()
- bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )

*Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.*

#### 6.356.1 Detailed Description

An **Executor** (p.1869) that provides methods to manage termination and methods that can produce a **Future** (p.1966) for tracking progress of one or more asynchronous tasks. An **ExecutorService** (p.1871) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p.1871). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p.1871) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p.1870)(decaf.lang Runnable (p.3325)) by creating and returning a **Future** (p.1966) that can be used to cancel execution and/or wait for completion. Methods invokeAny and invokeAll perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class ExecutorCompletionService can be used to write customized variants of these methods.)

The Executors class provides factory methods for the executor services provided in this package.

Since:

1.0

## 6.356.2 Constructor & Destructor Documentation

**6.356.2.1** `virtual decaf::util::concurrent::ExecutorService::~~ExecutorService ()`  
[inline, virtual]

## 6.356.3 Member Function Documentation

**6.356.3.1** `bool decaf::util::concurrent::ExecutorService::awaitTermination`  
(long long *timeout*, const TimeUnit & *unit*) throw (  
decaf::lang::exceptions::InterruptedException ) [pure virtual]

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

### Parameters:

*timeout* The amount of time to wait before timing out the Wait operation.

*unit* The Units that comprise the timeout value.

### Returns:

true if the executor terminated before the given timeout value elapsed.

### Exceptions:

*InterruptedException* - if interrupted while waiting.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutorService.h`



## 6.357 activemq::transport::failover::FailoverTransport Class Reference

#include <src/main/activemq/transport/failover/FailoverTransport.h> Inheritance diagram for activemq::transport::failover::FailoverTransport:

### Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()  
*Indicates that the **Transport** (p. 3883) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)  
*Adds a New URI to the List of URIs this **transport** (p. 97) can Connect to.*
- virtual void **addURI** (const **List**< **URI** > &uris)  
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3883) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)  
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3883) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3883) should result in that **Transport** (p. 3883) being disposed of.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87).*
- virtual void **stop** () throw ( decaf::io::IOException )  
*Stop the **Transport** (p. 3883).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP\_UNUSED)  
*Sets the WireFormat instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous events from this **transport** (p. 97).*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3883) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3883) been shutdown and no longer usable.*
- bool **isInitialized** () const  
*Returns true if the **Transport** (p. 3883) has been initialized by a BrokerInfo command.*
- void **setInitialized** (bool value)  
*Sets the initialized **state** (p. 95) of this **Transport** (p. 3883) to true.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeId)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()  
*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1873), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw ( **decaf::io::IOException** )  
*reconnect to another location*
- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const

- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- int **getStartupMaxReconnectAttempts** () const
- void **setStartupMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)
- void **setConnectionInterruptProcessingComplete** (const **Pointer**< **commands::ConnectionId** > &connectionId)

## Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw ( decaf::io::IOException )  
*Given a **Transport** (p. 3883) restore the **state** (p. 95) of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const decaf::lang::Exception &error) throw ( decaf::lang::Exception )  
*Called when this class' **TransportListener** (p. 3900) is notified of a Failure.*

## Friends

- class **FailoverTransportListener**

## 6.357.1 Constructor & Destructor Documentation

**6.357.1.1** **activemq::transport::failover::FailoverTransport::FailoverTransport** ()

**6.357.1.2** **virtual**  
**activemq::transport::failover::FailoverTransport::~~FailoverTransport** ()  
 [virtual]

## 6.357.2 Member Function Documentation

**6.357.2.1** **void** **activemq::transport::failover::FailoverTransport::add** (const std::string & uri)

Adds a New URI to the List of URIs this **transport** (p. 97) can Connect to.

**Parameters:**

*uri* A String version of a URI to add to the URIs to **failover** (p. 99) to.

**6.357.2.2 virtual void activemq::transport::failover::FailoverTransport::addURI  
(const List< URI > & *uris*) [virtual]**

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3883) is a composite of.

**Parameters:**

*uris* The new URIs to add to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 1226).

**6.357.2.3 virtual void activemq::transport::failover::FailoverTransport::close ()  
throw ( decaf::io::IOException ) [virtual]**

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions:**

*IOException* if errors occur.

Implements **decaf::io::Closeable** (p. 1149).

- 6.357.2.4** `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier () const [inline]`
- 6.357.2.5** `int activemq::transport::failover::FailoverTransport::getBackupPoolSize () const [inline]`
- 6.357.2.6** `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay () const [inline]`
- 6.357.2.7** `int activemq::transport::failover::FailoverTransport::getMaxCacheSize () const [inline]`
- 6.357.2.8** `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts () const [inline]`
- 6.357.2.9** `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay () const [inline]`
- 6.357.2.10** `long long activemq::transport::failover::FailoverTransport::getReconnectDelay () const [inline]`
- 6.357.2.11** `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress () const [virtual]`

**Returns:**

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3884).

- 6.357.2.12** `int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts () const [inline]`
- 6.357.2.13** `long long activemq::transport::failover::FailoverTransport::getTimeout () const [inline]`
- 6.357.2.14** `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener () const [virtual]`

Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).

**Returns:**

The listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3884).

**6.357.2.15**    `void activemq::transport::failover::FailoverTransport::handleTransportFailure (const decaf::lang::Exception & error) throw ( decaf::lang::Exception ) [protected]`

Called when this class' **TransportListener** (p. 3900) is notified of a Failure.

**Parameters:**

*error* - The CMS Exception that was thrown.

**Exceptions:**

*Exception* if an error occurs.

**6.357.2.16**    `bool activemq::transport::failover::FailoverTransport::isBackup () const [inline]`

**6.357.2.17**    `virtual bool activemq::transport::failover::FailoverTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3883) been shutdown and no longer usable.

**Returns:**

true if the **Transport** (p. 3883)

Implements **activemq::transport::Transport** (p. 3885).

**6.357.2.18**    `virtual bool activemq::transport::failover::FailoverTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3883) Connected to its Broker.

**Returns:**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3885).

**6.357.2.19**    `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns:**

true if the **Transport** (p. 3883) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3885).

**6.357.2.20** `bool activemq::transport::failover::FailoverTransport::isInitialized () const [inline]`

Returns true if the **Transport** (p. 3883) has been initialized by a **BrokerInfo** command.

**Returns:**

true if the **Transport** (p. 3883) has been initialized by a **BrokerInfo** command.

**6.357.2.21** `virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]`

**Returns:**

true if there is a need for the `iterate` method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1221).

**6.357.2.22** `bool activemq::transport::failover::FailoverTransport::isRandomize () const [inline]`

**6.357.2.23** `bool activemq::transport::failover::FailoverTransport::isTrackMessages () const [inline]`

**6.357.2.24** `bool activemq::transport::failover::FailoverTransport::isTrackTransactionProducers () const [inline]`

**6.357.2.25** `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff () const [inline]`

**6.357.2.26** `virtual bool activemq::transport::failover::FailoverTransport::iterate () [virtual]`

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1873), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

**Returns:**

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 3734).

**6.357.2.27** `virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.

**Parameters:**

*typeId* - The type\_info of the Object we are searching for.

**Returns:**

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p.3885).

References **activemq::transport::Transport::narrow()**.

**6.357.2.28** **virtual void activemq::transport::failover::FailoverTransport::oneway**  
**(const Pointer< Command > & command)**  
**throw ( decaf::io::IOException, de-**  
**cafe::lang::exceptions::UnsupportedOperationException )**  
**[virtual]**

Sends a one-way command. Does not wait for any response from the broker.

**Parameters:**

*command* the command to be sent.

**Exceptions:**

**IOException** if an exception occurs during writing of the command.

**UnsupportedOperationException** if this method is not implemented by this **transport**  
 (p. 97).

Implements **activemq::transport::Transport** (p.3886).

**6.357.2.29** **virtual void activemq::transport::failover::FailoverTransport::reconnect**  
**(const decaf::net::URI & uri) throw ( decaf::io::IOException )**  
**[virtual]**

reconnect to another location

**Parameters:**

*uri*

**Exceptions:**

**IOException** on failure of if not supported

Implements **activemq::transport::Transport** (p.3886).

**6.357.2.30** **void activemq::transport::failover::FailoverTransport::reconnect ( )**

Indicates that the **Transport** (p.3883) needs to reconnect to another URI in its list.



**6.357.2.31 virtual void activemq::transport::failover::FailoverTransport::removeURI (const List< URI > & *uris*) [virtual]**

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3883) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3883) should result in that **Transport** (p. 3883) being disposed of.

**Parameters:**

*uris* The new URIs to remove to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 1227).

**6.357.2.32 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]**

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* - The command to be sent.

*timeout* - The time to wait for this response.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3886).

**6.357.2.33 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]**

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* the command to be sent.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3887).

**6.357.2.34** `void activemq::transport::failover::FailoverTransport::restoreTransport (const Pointer< Transport > & transport) throw ( decaf::io::IOException )` [protected]

Given a **Transport** (p. 3883) restore the **state** (p. 95) of the Client's connection to the Broker using the data accumulated in the State Tracker.

**Parameters:**

*transport* (p. 97) The new **Transport** (p. 3883) connected to the Broker.

**Exceptions:**

*IOException* if an errors occurs while restoring the old **state** (p. 95).

**6.357.2.35** `void activemq::transport::failover::FailoverTransport::setBackOffMultiplier (long long value)` [inline]

**6.357.2.36** `void activemq::transport::failover::FailoverTransport::setBackup (bool value)` [inline]

**6.357.2.37** `void activemq::transport::failover::FailoverTransport::setBackupPoolSize (int value)` [inline]

**6.357.2.38** `void activemq::transport::failover::FailoverTransport::setConnectionInterruptProcessingComplete (const Pointer< commands::ConnectionId > & connectionId)`

**6.357.2.39** `void activemq::transport::failover::FailoverTransport::setInitialized (bool value)` [inline]

Sets the initialized **state** (p. 95) of this **Transport** (p. 3883) to true.

**Parameters:**

*value* - true if this **Transport** (p. 3883) has been initialized.

- 6.357.2.40 void `activemq::transport::failover::FailoverTransport::setInitialReconnectDelay` (long long *value*) [inline]
- 6.357.2.41 void `activemq::transport::failover::FailoverTransport::setMaxCacheSize` (int *value*) [inline]
- 6.357.2.42 void `activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts` (int *value*) [inline]
- 6.357.2.43 void `activemq::transport::failover::FailoverTransport::setMaxReconnectDelay` (long long *value*) [inline]
- 6.357.2.44 void `activemq::transport::failover::FailoverTransport::setRandomize` (bool *value*) [inline]
- 6.357.2.45 void `activemq::transport::failover::FailoverTransport::setReconnectDelay` (long long *value*) [inline]
- 6.357.2.46 void `activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts` (int *value*) [inline]
- 6.357.2.47 void `activemq::transport::failover::FailoverTransport::setTimeout` (long long *value*) [inline]
- 6.357.2.48 void `activemq::transport::failover::FailoverTransport::setTrackMessages` (bool *value*) [inline]
- 6.357.2.49 void `activemq::transport::failover::FailoverTransport::setTrackTransactionProducers` (bool *value*) [inline]
- 6.357.2.50 virtual void `activemq::transport::failover::FailoverTransport::setTransportListener` (TransportListener \* *listener*) [virtual]

Sets the observer of asynchronous events from this **transport** (p. 97).

#### Parameters:

*listener* the listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3887).

**6.357.2.51** void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool *value*) [inline]

**6.357.2.52** virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP\_UNUSED*) [inline, virtual]

Sets the WireFormat instance to use.

#### Parameters:

*wireFormat* The WireFormat the object used to encode / decode **commands** (p.87).

**6.357.2.53** virtual void activemq::transport::failover::FailoverTransport::start () throw ( decaf::io::IOException ) [virtual]

Starts this **transport** (p.97) object and creates the thread for polling on the input stream for **commands** (p.87). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions:

*IOException* if an error occurs or if this **transport** (p.97) has already been closed.

Implements **activemq::transport::Transport** (p.3888).

**6.357.2.54** virtual void activemq::transport::failover::FailoverTransport::stop () throw ( decaf::io::IOException ) [virtual]

Stop the **Transport** (p.3883).

#### Exceptions:

*IOException* if an error occurs while stopping the **Transport** (p.3883).

Implements **activemq::transport::Transport** (p.3888).

## 6.357.3 Friends And Related Function Documentation

**6.357.3.1** friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

## 6.358 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a **FailoverTransport** (p.1873).

#include <src/main/activemq/transport/failover/FailoverTransportFactory.h> Inheritance diagram for activemq::transport::failover::FailoverTransportFactory:

### Public Member Functions

- virtual **~FailoverTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )

*Creates a fully configured **Transport** (p.3883) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )

*Creates a slimmed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.*

### Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties) throw ( exceptions::ActiveMQException )

*Creates a slimmed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.*

#### 6.358.1 Detailed Description

Creates an instance of a **FailoverTransport** (p.1873).

Since:

3.0

## 6.358.2 Constructor & Destructor Documentation

**6.358.2.1** virtual  
 activemq::transport::failover::FailoverTransportFactory::~~FailoverTransportFactory  
 () [inline, virtual]

## 6.358.3 Member Function Documentation

**6.358.3.1** virtual Pointer<Transport> ac-  
 tivemq::transport::failover::FailoverTransportFactory::create (const  
 decaf::net::URI & *location*) throw ( exceptions::ActiveMQException )  
 [virtual]

Creates a fully configured **Transport** (p.3883) instance which could be a chain of filters and transports.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements **activemq::transport::TransportFactory** (p.3889).

**6.358.3.2** virtual Pointer<Transport> ac-  
 tivemq::transport::failover::FailoverTransportFactory::createComposite  
 (const decaf::net::URI & *location*) throw ( excep-  
 tions::ActiveMQException ) [virtual]

Creates a slimed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements **activemq::transport::TransportFactory** (p.3890).

**6.358.3.3** virtual Pointer<Transport> ac-  
 tivemq::transport::failover::FailoverTransportFactory::doCreateComposite  
 (const decaf::net::URI & *location*, const decaf::util::Properties &  
*properties*) throw ( exceptions::ActiveMQException ) [protected,  
 virtual]

Creates a slimed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.

**Parameters:**

*location* - URI location to connect to.

*properties* - Properties to apply to the **transport** (p. 97).

**Returns:**

Pointer to a new **FailoverTransport** (p. 1873) instance.

**Exceptions:**

*ActiveMQException* if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

## 6.359 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 3883) to perform the work of responding to events from the active **Transport** (p. 3883).

#include <src/main/activemq/transport/failover/FailoverTransportListener.h> Inheritance diagram for activemq::transport::failover::FailoverTransportListener:

### Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** \*parent)
- virtual ~**FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

*Event handler for the receipt of a command.*

- virtual void **onException** (const **decaf::lang::Exception** &ex)

*Event handler for an exception from a command **transport** (p. 97).*

- virtual void **transportInterrupted** ()

*The **transport** (p. 97) has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()

*The **transport** (p. 97) has resumed after an interruption.*

### 6.359.1 Detailed Description

Utility class used by the **Transport** (p. 3883) to perform the work of responding to events from the active **Transport** (p. 3883).

**Since:**

3.0



## 6.359.2 Constructor & Destructor Documentation

- 6.359.2.1** `activemq::transport::failover::FailoverTransportListener::FailoverTransportListener(FailoverTransport * parent)`
- 6.359.2.2** `virtual  
activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener()  
()` [virtual]

## 6.359.3 Member Function Documentation

- 6.359.3.1** `virtual void ac-  
tivemq::transport::failover::FailoverTransportListener::onCommand  
(const Pointer< Command > & command)` [virtual]

Event handler for the receipt of a command. The **transport** (p. 97) passes off all received **commands** (p. 87) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3883) deletes the command upon receipt.

### Parameters:

*command* the received command object.

Implements **activemq::transport::TransportListener** (p. 3900).

- 6.359.3.2** `virtual void ac-  
tivemq::transport::failover::FailoverTransportListener::onException  
(const decaf::lang::Exception & ex)` [virtual]

Event handler for an exception from a command **transport** (p. 97).

### Parameters:

*ex* The exception.

Implements **activemq::transport::TransportListener** (p. 3901).

- 6.359.3.3** `virtual void ac-  
tivemq::transport::failover::FailoverTransportListener::transportInterrupted  
()` [virtual]

The **transport** (p. 97) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3901).

- 6.359.3.4** `virtual void ac-  
tivemq::transport::failover::FailoverTransportListener::transportResumed  
()` [virtual]

The **transport** (p. 97) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3901).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

## 6.360 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

#include <src/main/decaf/io/FileDescriptor.h>Inheritance diagram for decaf::io::FileDescriptor:

### Public Member Functions

- **FileDescriptor** ()
- virtual **~FileDescriptor** ()
- void **sync** ()  
*Force any/all buffered data for this **FileDescriptor** (p. 1891) to be flushed to the underlying device.*
- bool **valid** ()  
*Indicates whether the File Descriptor is valid.*

### Static Public Attributes

- static **FileDescriptor in**  
*A handle to the standard input stream.*
- static **FileDescriptor out**  
*A handle to the standard output stream.*
- static **FileDescriptor err**  
*A handle to the standard error stream.*

### Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

### Protected Attributes

- long **descriptor**
- bool **readonly**

#### 6.360.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

Since:

1.0

## 6.360.2 Constructor & Destructor Documentation

**6.360.2.1** `decaf::io::FileDescriptor::FileDescriptor (long value, bool readonly)` [protected]

**6.360.2.2** `decaf::io::FileDescriptor::FileDescriptor ()`

**6.360.2.3** `virtual decaf::io::FileDescriptor::~~FileDescriptor ()` [virtual]

## 6.360.3 Member Function Documentation

**6.360.3.1** `void decaf::io::FileDescriptor::sync ()`

Force any/all buffered data for this **FileDescriptor** (p.1891) to be flushed to the underlying device. This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p.940) the stream must first be flushed before this method can force the data to be sent to the output device.

**6.360.3.2** `bool decaf::io::FileDescriptor::valid ()`

Indicates whether the File Descriptor is valid.

### Returns:

true for a valid descriptor such as open socket or file, false otherwise.

## 6.360.4 Field Documentation

**6.360.4.1** `long decaf::io::FileDescriptor::descriptor` [protected]

**6.360.4.2** `FileDescriptor decaf::io::FileDescriptor::err` [static]

A handle to the standard error stream. Usually, this file descriptor is not used directly, but rather via the output stream known as `System::err`.

**6.360.4.3** `FileDescriptor decaf::io::FileDescriptor::in` [static]

A handle to the standard input stream. Usually, this file descriptor is not used directly, but rather via the input stream known as `System::in`.

**6.360.4.4** `FileDescriptor decaf::io::FileDescriptor::out` [static]

A handle to the standard output stream. Usually, this file descriptor is not used directly, but rather via the output stream known as `System::out`.

**6.360.4.5** `bool decaf::io::FileDescriptor::readonly` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FileDescriptor.h`

## 6.361 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1893) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

### Public Member Functions

- virtual **~Filter** ()
- virtual bool **isLoggable** (const **LogRecord** &record) const =0

*Check if a given log record should be published.*

#### 6.361.1 Detailed Description

A **Filter** (p.1893) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p.2386) and each **Handler** (p.1978) can have a filter associated with it. The **Logger** (p.2386) or **Handler** (p.1978) will call the **isLoggable** method to check if a given **LogRecord** (p.2413) should be published. If **isLoggable** returns false, the **LogRecord** (p.2413) will be discarded.

#### 6.361.2 Constructor & Destructor Documentation

**6.361.2.1** virtual decaf::util::logging::Filter::~Filter () [inline, virtual]

#### 6.361.3 Member Function Documentation

**6.361.3.1** virtual bool decaf::util::logging::Filter::isLoggable (const **LogRecord** &*record*) const [pure virtual]

Check if a given log record should be published.

##### Parameters:

*record* the **LogRecord** (p.2413) to check.

##### Returns:

true if the record is loggable.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Filter.h**

## 6.362 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p.1894) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

#include <src/main/decaf/io/FilterInputStream.h> Inheritance diagram for decaf::io::FilterInputStream:

### Public Member Functions

- **FilterInputStream** (**InputStream** \*inputStream, bool own=false)

*Constructor to create a wrapped **InputStream** (p. 2043).*

- virtual ~**FilterInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

**Returns:**

*the number of bytes available on this input stream.*

**Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.*

*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters:**

*num The number of bytes to skip.*

**Returns:**

*total bytes skipped*

**Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters:**

**readLimit** *The max bytes read before marked position is invalid.*

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2142).*

**Exceptions:**

**IOException** (p. 2142) *if an I/O error occurs.*

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns:**

*true if this stream instance supports marks*

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual bool **isClosed** () const

## Protected Attributes

- **InputStream \* inputStream**

- bool **own**
- volatile bool **closed**

### 6.362.1 Detailed Description

A **FilterInputStream** (p.1894) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p.1894) itself simply overrides all methods of **InputStream** (p.2043) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p.1894) may further override some of these methods and may also provide additional methods and fields.

### 6.362.2 Constructor & Destructor Documentation

#### 6.362.2.1 `decaf::io::FilterInputStream::FilterInputStream (InputStream * inputStream, bool own = false)`

Constructor to create a wrapped **InputStream** (p.2043).

##### Parameters:

- inputStream* The stream to wrap and filter.
- own* Indicates if we own the stream object, defaults to false.

#### 6.362.2.2 `virtual decaf::io::FilterInputStream::~~FilterInputStream ()` [virtual]

### 6.362.3 Member Function Documentation

#### 6.362.3.1 `virtual int decaf::io::FilterInputStream::available () const throw (decaf::io::IOException)` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

##### Returns:

- the number of bytes available on this input stream.

##### Exceptions:

- IOException* (p.2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p.2045).

Reimplemented in **decaf::io::BufferedInputStream** (p.936), **decaf::io::PushbackInputStream** (p.3143), and **decaf::util::zip::InflaterInputStream** (p.2038).



### 6.362.3.2 virtual void decaf::io::FilterInputStream::close () throw ( decaf::io::IOException ) [virtual]

Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2045).

Reimplemented in **decaf::io::BufferedInputStream** (p. 937), and **decaf::util::zip::InflaterInputStream** (p. 2039).

### 6.362.3.3 virtual int decaf::io::FilterInputStream::doReadArray (unsigned char \* *buffer*, int *size*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2045).

### 6.362.3.4 virtual int decaf::io::FilterInputStream::doReadArrayBounded (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2046).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2399), **decaf::io::BufferedInputStream** (p. 937), **decaf::io::PushbackInputStream** (p. 3144), **decaf::util::zip::CheckedInputStream** (p. 1138), and **decaf::util::zip::InflaterInputStream** (p. 2039).

### 6.362.3.5 virtual int decaf::io::FilterInputStream::doReadByte () throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 2046).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2400), **decaf::io::BufferedInputStream** (p. 937), **decaf::io::PushbackInputStream** (p. 3144), **decaf::util::zip::CheckedInputStream** (p. 1138), and **decaf::util::zip::InflaterInputStream** (p. 2039).

### 6.362.3.6 virtual bool decaf::io::FilterInputStream::isClosed () const [protected, virtual]

#### Returns:

true if this stream has been closed.

### 6.362.3.7 virtual void decaf::io::FilterInputStream::mark (int *readLimit*) [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters:

***readLimit*** The max bytes read before marked position is invalid.

Reimplemented from **`decaf::io::InputStream`** (p. 2046).

Reimplemented in **`decaf::io::BufferedInputStream`** (p. 937), **`decaf::io::PushbackInputStream`** (p. 3144), and **`decaf::util::zip::InflaterInputStream`** (p. 2039).

#### 6.362.3.8 **`virtual bool decaf::io::FilterInputStream::markSupported () const`** [virtual]

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns `false`.

#### Returns:

`true` if this stream instance supports marks

Reimplemented from **`decaf::io::InputStream`** (p. 2047).

Reimplemented in **`decaf::io::BufferedInputStream`** (p. 937), **`decaf::io::PushbackInputStream`** (p. 3144), and **`decaf::util::zip::InflaterInputStream`** (p. 2040).

#### 6.362.3.9 **`virtual void decaf::io::FilterInputStream::reset () throw ( decaf::io::IOException )`** [virtual]

Repositions this stream to the position at the time the `mark` method was last called on this input stream.

If the method `markSupported` returns `true`, then: \* If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **`IOException`** (p. 2142) might be thrown. \* If such an **`IOException`** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns `false`, then: \* The call to `reset` may throw an **`IOException`** (p. 2142). \* If an **`IOException`** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **`IOException`** (p. 2142).

**Exceptions:**

***IOException*** (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2049).

Reimplemented in **decaf::io::BufferedInputStream** (p. 938), **decaf::io::PushbackInputStream** (p. 3145), and **decaf::util::zip::InflaterInputStream** (p. 2040).

**6.362.3.10 virtual long long decaf::io::FilterInputStream::skip  
(long long num) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]**

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters:**

*num* The number of bytes to skip.

**Returns:**

total bytes skipped

**Exceptions:**

***IOException*** (p. 2142) if an I/O error occurs.

***UnsupportedOperationException*** if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2050).

Reimplemented in **decaf::io::BufferedInputStream** (p. 938), **decaf::io::PushbackInputStream** (p. 3145), **decaf::util::zip::CheckedInputStream** (p. 1139), and **decaf::util::zip::InflaterInputStream** (p. 2041).

**6.362.4 Field Documentation**

**6.362.4.1 volatile bool decaf::io::FilterInputStream::closed [protected]**

**6.362.4.2 InputStream\* decaf::io::FilterInputStream::inputStream [protected]**

**6.362.4.3 bool decaf::io::FilterInputStream::own [protected]**

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterInputStream.h**

## 6.363 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

#include <src/main/decaf/io/FilterOutputStream.h> Inheritance diagram for decaf::io::FilterOutputStream:

### Public Member Functions

- **FilterOutputStream (OutputStream \*outputStream, bool own=false)**  
*Constructor, creates a wrapped output stream.*
- virtual ~**FilterOutputStream ()**
- virtual void **flush ()** throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*  
**Exceptions:**  
*IOException (p. 2142) if an I/O error occurs.*  
*The default implementation of this method does nothing.*
- virtual void **close ()** throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*  
*The object is generally no longer usable after calling close.*  
**Exceptions:**  
*IOException (p. 2142) if an error occurs while closing.*  
*The default implementation of this method does nothing.*
- virtual std::string **toString ()** const  
*Output a String representation of this object.*  
*The default version of this method just prints the Class Name.*  
**Returns:**  
*a string representation of the object.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual bool **isClosed ()** const

### Protected Attributes

- **OutputStream \* outputStream**
- **bool own**
- **volatile bool closed**

### 6.363.1 Detailed Description

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1900) itself simply overrides all methods of **OutputStream** (p. 2907) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1900) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2043) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataOutputStream (p. 1579) os = new DataOutputStream (p. 1579)( new Output-Stream() (p. 2909), true )
```

### 6.363.2 Constructor & Destructor Documentation

#### 6.363.2.1 decaf::io::FilterOutputStream::FilterOutputStream (OutputStream \* *outputStream*, bool *own* = false)

Constructor, creates a wrapped output stream.

Parameters:

*outputStream* the **OutputStream** (p. 2907) to wrap

*own* If true, this object will control the lifetime of the output stream that it encapsulates.

#### 6.363.2.2 virtual decaf::io::FilterOutputStream::~~FilterOutputStream () [virtual]

### 6.363.3 Member Function Documentation

#### 6.363.3.1 virtual void decaf::io::FilterOutputStream::close () throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions:

**IOException** (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing. The close method of **FilterOutputStream** (p. 1900) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2909).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1718).

**6.363.3.2** `virtual void decaf::io::FilterOutputStream::doWriteArray (const unsigned char * buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2909).

Reimplemented in `decaf::io::BufferedOutputStream` (p. 941).

**6.363.3.3** `virtual void decaf::io::FilterOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2909).

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2401), `decaf::io::BufferedOutputStream` (p. 941), `decaf::io::DataOutputStream` (p. 1580), `decaf::util::zip::CheckedOutputStream` (p. 1141), and `decaf::util::zip::DeflaterOutputStream` (p. 1719).

**6.363.3.4** `virtual void decaf::io::FilterOutputStream::doWriteByte (unsigned char value) throw ( decaf::io::IOException )` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2910).

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2402), `decaf::io::BufferedOutputStream` (p. 941), `decaf::io::DataOutputStream` (p. 1581), `decaf::util::zip::CheckedOutputStream` (p. 1141), and `decaf::util::zip::DeflaterOutputStream` (p. 1719).

**6.363.3.5** `virtual void decaf::io::FilterOutputStream::flush () throw ( decaf::io::IOException )` [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

The default implementation of this method does nothing. The flush method of **FilterOutputStream** (p. 1900) calls the flush method of its underlying output stream.

Reimplemented from `decaf::io::OutputStream` (p. 2910).

Reimplemented in `decaf::io::BufferedOutputStream` (p. 941).

**6.363.3.6** `virtual bool decaf::io::FilterOutputStream::isClosed () const` [protected, virtual]

#### Returns:

true if this stream has been closed.

### 6.363.3.7 virtual std::string decaf::io::FilterOutputStream::toString () const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns:

a string representation of the object.

The toString method of **FilterOutputStream** (p. 1900) calls the toString method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2911).

## 6.363.4 Field Documentation

### 6.363.4.1 volatile bool decaf::io::FilterOutputStream::closed [protected]

### 6.363.4.2 OutputStream\* decaf::io::FilterOutputStream::outputStream [protected]

### 6.363.4.3 bool decaf::io::FilterOutputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterOutputStream.h**

## 6.364 decaf::lang::Float Class Reference

#include <src/main/decaf/lang/Float.h> Inheritance diagram for decaf::lang::Float:

### Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const  
*Compares this **Float** (p. 1904) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Float** &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const float &f) const  
*Compares this **Float** (p. 1904) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const float &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*



- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*
- bool **isInfinite** () const
- bool **isNaN** () const

## Static Public Member Functions

- static int **compare** (float f1, float f2)  
*Compares the two specified double values.*
- static int **floatToIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.*
- static int **floatToRawIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.*
- static float **intBitsToFloat** (int bits)  
*Returns the float value corresponding to a given bit representation.*
- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1904).*
- static std::string **toHexString** (float value)  
*Returns a hexadecimal string representation of the float argument.*
- static std::string **toString** (float value)  
*Returns a string representation of the float argument.*
- static **Float** **valueOf** (float value)  
*Returns a **Float** (p. 1904) instance representing the specified float value.*
- static **Float** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Float** (p. 1904) instance that wraps a primitive float which is parsed from the string value passed.*

## Static Public Attributes

- static const int **SIZE** = 32  
*The size in bits of the primitive int type.*
- static const float **MAX\_VALUE**

*The maximum value that the primitive type can hold.*

- static const float **MIN\_VALUE**

*The minimum value that the primitive type can hold.*

- static const float **NaN**

*Constant for the Not a **Number** (p. 2835) Value.*

- static const float **POSITIVE\_INFINITY**

*Constant for Positive Infinity.*

- static const float **NEGATIVE\_INFINITY**

*Constant for Negative Infinity.*

### 6.364.1 Constructor & Destructor Documentation

#### 6.364.1.1 `decaf::lang::Float::Float (float value)`

##### Parameters:

*value* - the primitive type to wrap

#### 6.364.1.2 `decaf::lang::Float::Float (double value)`

##### Parameters:

*value* - the primitive type to wrap

#### 6.364.1.3 `decaf::lang::Float::Float (const std::string & value) throw ( exceptions::NumberFormatException )`

##### Parameters:

*value* - the string to convert to a primitive type to wrap

#### 6.364.1.4 `virtual decaf::lang::Float::~~Float () [inline, virtual]`

### 6.364.2 Member Function Documentation

#### 6.364.2.1 `virtual unsigned char decaf::lang::Float::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

##### Returns:

byte the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2835).

**6.364.2.2 static int decaf::lang::Float::compare (float *f1*, float *f2*) [static]**

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float( f1 ).compareTo( Float( f2 ) )`

**Parameters:**

*f1* - the first double to compare

*f2* - the second double to compare

**Returns:**

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

**6.364.2.3 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]**

Compares this **Float** (p. 1904) instance with another.

**Parameters:**

*f* - the **Float** (p. 1904) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< float >** (p. 1214).

**6.364.2.4 virtual int decaf::lang::Float::compareTo (const Float & *f*) const [virtual]**

Compares this **Float** (p. 1904) instance with another.

**Parameters:**

*f* - the **Float** (p. 1904) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.364.2.5 virtual double decaf::lang::Float::doubleValue () const [inline, virtual]**

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.364.2.6** `bool decaf::lang::Float::equals (const float & f) const` [inline, virtual]**Parameters:**

*f* - the **Float** (p. 1904) object to compare against.

**Returns:**

true if the two **Float** (p. 1904) Objects have the same value.

Implements **decaf::lang::Comparable**< float > (p. 1215).

**6.364.2.7** `bool decaf::lang::Float::equals (const Float & f) const` [inline]**Parameters:**

*f* - the **Float** (p. 1904) object to compare against.

**Returns:**

true if the two **Float** (p. 1904) Objects have the same value.

**6.364.2.8** `static int decaf::lang::Float::floatToIntBits (float value)` [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7c000000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1909) method, will produce a floating-point value the same as the argument to floatToIntBits (except all NaN values are collapsed to a single "canonical" NaN value).

**Parameters:**

*value* - the float to convert to int bits

**Returns:**

the int that holds the float's value

**6.364.2.9** `static int decaf::lang::Float::floatToRawIntBits (float value)` [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the `floatToIntBits` method, `intToRawIntBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the `intBitsToFloat(int)` (p. 1909) method, will produce a floating-point value the same as the argument to `floatToRawIntBits`.

#### Parameters:

*value* The float to convert to a raw int.

#### Returns:

the raw int value of the float

#### 6.364.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

#### Returns:

float the value of the receiver.

Implements `decaf::lang::Number` (p. 2836).

#### 6.364.2.11 static float decaf::lang::Float::intBitsToFloat (int bits) [static]

Returns the float value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the `Float::floatToRawIntBits` (p. 1908) method.

#### Parameters:

*bits* - the bits of the float encoded as a float

#### Returns:

a new float created from the int bits.

#### 6.364.2.12 virtual int decaf::lang::Float::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

#### Returns:

int the value of the receiver.

Implements `decaf::lang::Number` (p. 2836).

**6.364.2.13**   `static bool decaf::lang::Float::isInfinite (float value)`   [static]

**Parameters:**

*value* - The float to check.

**Returns:**

true if the float is equal to infinity.

**6.364.2.14**   `bool decaf::lang::Float::isInfinite () const`

**Returns:**

true if the float is equal to positive infinity.

**6.364.2.15**   `static bool decaf::lang::Float::isNaN (float value)`   [static]

**Parameters:**

*value* - The float to check.

**Returns:**

true if the float is equal to NaN.

**6.364.2.16**   `bool decaf::lang::Float::isNaN () const`

**Returns:**

true if the float is equal to NaN.

**6.364.2.17**   `virtual long long decaf::lang::Float::longValue () const`   [inline, virtual]

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.364.2.18**   `virtual bool decaf::lang::Float::operator< (const float & f) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*f* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< float > (p. 1215).

**6.364.2.19**    **virtual bool decaf::lang::Float::operator< (const Float & f) const**  
                 [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*f* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.364.2.20**    **virtual bool decaf::lang::Float::operator== (const float & f) const**  
                 [inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*f* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< float > (p. 1215).

**6.364.2.21**    **virtual bool decaf::lang::Float::operator== (const Float & f) const**  
                 [inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*f* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.364.2.22**    **static float decaf::lang::Float::parseFloat (const std::string & value)**  
                 **throw ( exceptions::NumberFormatException )** [static]

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1904).

**Parameters:**

*value* - the string to parse

**Returns:**

a float parsed from the string

**Exceptions:**

*NumberFormatException*

**6.364.2.23** `virtual short decaf::lang::Float::shortValue () const` `[inline, virtual]`

Answers the short value which the receiver represents.

**Returns:**

short the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2837).

**6.364.2.24** `static std::string decaf::lang::Float::toHexString (float value)` `[static]`

Returns a hexadecimal string representation of the float argument. All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 2089) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

**Parameters:**

*value* - The float to convert to a string

**Returns:**

the Hex formatted float string.



**6.364.2.25 static std::string decaf::lang::Float::toString (float *value*) [static]**

Returns a string representation of the float argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude *m*:  
o If *m* is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".  
o If *m* is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".  
o If *m* is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of *m*, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of *m*.  
o If *m* is less than 10<sup>-3</sup> or greater than or equal to 10<sup>7</sup>, then it is represented in so-called "computerized scientific notation." Let *n* be the unique integer such that 10<sup>*n*</sup> ≤ *m* < 10<sup>*n*+1</sup>; then let *a* be the mathematically exact quotient of *m* and 10<sup>*n*</sup> so that 1 ≤ *a* < 10. The magnitude is then represented as the integer part of *a*, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of *a*, followed by the letter 'E', followed by a representation of *n* as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2089).

**Parameters:**

*value* - The float to convert to a string

**Returns:**

the formatted float string.

**6.364.2.26 std::string decaf::lang::Float::toString () const****Returns:**

this **Float** (p. 1904) Object as a **String** (p. 3665) Representation

**6.364.2.27 static Float decaf::lang::Float::valueOf (const std::string & *value*) throw ( exceptions::NumberFormatException ) [static]**

Returns a **Float** (p. 1904) instance that wraps a primitive float which is parsed from the string value passed.

**Parameters:**

*value* - the string to parse

**Returns:**

a new **Float** (p. 1904) instance wrapping the float parsed from value

**Exceptions:**

**NumberFormatException** on error.

**6.364.2.28**   `static Float decaf::lang::Float::valueOf (float value)`   [static]

Returns a **Float** (p. 1904) instance representing the specified float value.

**Parameters:**

*value* - float to wrap

**Returns:**

new **Float** (p. 1904) instance wrapping the primitive value

**6.364.3**   **Field Documentation****6.364.3.1**   `const float decaf::lang::Float::MAX_VALUE`   [static]

The maximum value that the primitive type can hold.

**6.364.3.2**   `const float decaf::lang::Float::MIN_VALUE`   [static]

The minimum value that the primitive type can hold.

**6.364.3.3**   `const float decaf::lang::Float::NaN`   [static]

Constant for the Not a **Number** (p. 2835) Value.

**6.364.3.4**   `const float decaf::lang::Float::NEGATIVE_INFINITY`   [static]

Constant for Negative Infinity.

**6.364.3.5**   `const float decaf::lang::Float::POSITIVE_INFINITY`   [static]

Constant for Positive Infinity.

**6.364.3.6**   `const int decaf::lang::Float::SIZE = 32`   [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

## 6.365 decaf::internal::nio::FloatArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/FloatArrayBuffer.h> Inheritance diagram for decaf::internal::nio::FloatArrayBuffer:

### Public Member Functions

- **FloatArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **FloatArrayBuffer** (p. 1915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **FloatArrayBuffer** (float \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **FloatArrayBuffer** (p. 1915) object that wraps the given array.*

- **FloatArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*

- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)

*Create a **FloatArrayBuffer** (p. 1915) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*

- virtual ~**FloatArrayBuffer** ()
- virtual float \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the float array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928).*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset into the backing array where index zero starts.*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual FloatBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only float buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only float buffer which the caller then owns.*

- virtual FloatBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns:**

*a reference to this **FloatBuffer** (p. 1925).*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual FloatBuffer \* **duplicate** ()

*Creates a new float buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

*The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*a new float **Buffer** (p. 928) which the caller owns.*

- virtual float **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns:**

*the float at the current position.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return.

- virtual float **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the float is to be read

**Returns:**

the float that is located at the given index

**Exceptions:**

**IndexOutOfBoundsException** if index is not smaller than the buffer's limit

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

- virtual FloatBuffer & **put** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The floats value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual FloatBuffer & **put** (int index, float value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given floats into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The floats to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written, or index is negative.  
**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual FloatBuffer \* **slice** () const

Creates a new **FloatBuffer** (p. 1925) whose content is a shared subsequence of this buffer's content.

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*the newly create **FloatBuffer** (p. 1925) which the caller owns.*

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **FloatArrayBuffer** (p. 1915) as Read-Only.*

### 6.365.1 Constructor & Destructor Documentation

**6.365.1.1** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (int size, bool readOnly = false) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **FloatArrayBuffer** (p.1915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

*size* The size of the array, this is the limit we read and write to.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

***IllegalArgumentException*** if the capacity value is negative.

**6.365.1.2** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * array, int size, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **FloatArrayBuffer** (p.1915) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The actual array to wrap.

*size* The size of the given array.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

**6.365.1.3** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer`  
`(const decaf::lang::Pointer< ByteArrayAdapter > &`  
`array, int offset, int capacity, bool readOnly = false)`  
`throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset. The capacity and limit of the new `FloatArrayBuffer` (p. 1915) will be that of the remaining capacity of the passed buffer.

**Parameters:**

*array* The `ByteArrayAdapter` to wrap.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

*NullPointerException* if array is NULL

*IndexOutOfBoundsException* if offset + length is greater than array size.

**6.365.1.4** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const`  
`FloatArrayBuffer & other)`

Create a `FloatArrayBuffer` (p. 1915) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

**Parameters:**

*other* The `FloatArrayBuffer` (p. 1915) this one is to mirror.

**6.365.1.5** `virtual decaf::internal::nio::FloatArrayBuffer::~~FloatArrayBuffer ()`  
`[virtual]`

**6.365.2 Member Function Documentation**

**6.365.2.1** `virtual float* decaf::internal::nio::FloatArrayBuffer::array ()`  
`throw ( decaf::lang::exceptions::UnsupportedOperationException,`  
`decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1928).

**6.365.2.2** `virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1928).

**6.365.2.3** `virtual FloatBuffer* de-  
caf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer ()  
const [virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1928).



**6.365.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ()  
throw ( decaf::nio::ReadOnlyBufferException ) [virtual]**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **FloatBuffer** (p. 1925).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1929).

**6.365.2.5 virtual FloatBuffer\* decaf::internal::nio::FloatArrayBuffer::duplicate ()  
[virtual]**

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new float **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1929).

**6.365.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get (int *index*) const  
throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]**

Absolute get method.

Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the float is to be read

**Returns:**

the float that is located at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit

Implements `decaf::nio::FloatBuffer` (p. 1930).

**6.365.2.7** `virtual float decaf::internal::nio::FloatArrayBuffer::get () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the float at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implements `decaf::nio::FloatBuffer` (p. 1931).

**6.365.2.8** `virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::FloatBuffer` (p. 1931).

**6.365.2.9** `virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements `decaf::nio::Buffer` (p. 931).

**6.365.2.10** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (int index, float value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given floats into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The floats to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1931).

**6.365.2.11** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The floats value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1932).

**6.365.2.12** `virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **FloatArrayBuffer** (p. 1915) as Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.365.2.13** `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice () const [virtual]`

Creates a new **FloatBuffer** (p. 1925) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **FloatBuffer** (p.1925) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p.1934).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/FloatArrayBuffer.h`

## 6.366 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:.

#include <src/main/decaf/nio/FloatBuffer.h> Inheritance diagram for decaf::nio::FloatBuffer:

### Public Member Functions

- virtual **~FloatBuffer** ()
- virtual std::string **toString** () const
- virtual float \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the float array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **FloatBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only float buffer that shares this buffer's content.*
- virtual **FloatBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **FloatBuffer** \* **duplicate** ()=0  
*Creates a new float buffer that shares this buffer's content.*
- virtual float **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual float **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **FloatBuffer** & **get** (std::vector< float > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **FloatBuffer** & **get** (float \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible float array.*
- **FloatBuffer** & **put** (**FloatBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the floats remaining in the given source buffer into this buffer.*

- **FloatBuffer** & **put** (const float \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers floats into this buffer from the given source array.*

- **FloatBuffer** & **put** (std::vector< float > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source floats array into this buffer.*

- virtual **FloatBuffer** & **put** (float value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given floats into this buffer at the current position, and then increments the position.*

- virtual **FloatBuffer** & **put** (int index, float value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given floats into this buffer at the given index.*

- virtual **FloatBuffer** \* **slice** () const =0

*Creates a new **FloatBuffer** (p. 1925) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **FloatBuffer** &value) const

- virtual bool **equals** (const **FloatBuffer** &value) const

- virtual bool **operator==** (const **FloatBuffer** &value) const

- virtual bool **operator<** (const **FloatBuffer** &value) const

## Static Public Member Functions

- static **FloatBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new Double buffer.*

- static **FloatBuffer** \* **wrap** (float \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **FloatBuffer** (p. 1925).*

- static **FloatBuffer** \* **wrap** (std::vector< float > &buffer)

*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1925).*

## Protected Member Functions

- **FloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **FloatBuffer** (p. 1925) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.366.1 Detailed Description

This class defines four categories of operations upon float buffers:
 

- o Absolute and relative get and put methods that read and write single floats;
- o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer
- o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.366.2 Constructor & Destructor Documentation

#### 6.366.2.1 decaf::nio::FloatBuffer::FloatBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **FloatBuffer** (p. 1925) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

##### Parameters:

*capacity* The size and limit of the **Buffer** (p. 928) in floats.

##### Exceptions:

*IllegalArgumentException* if capacity is negative.

#### 6.366.2.2 virtual decaf::nio::FloatBuffer::~~FloatBuffer () [inline, virtual]

### 6.366.3 Member Function Documentation

#### 6.366.3.1 static FloatBuffer\* decaf::nio::FloatBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

##### Parameters:

*capacity* The size of the Double buffer in floats.

##### Returns:

the **FloatBuffer** (p. 1925) that was allocated, caller owns.

**6.366.3.2** `virtual float* decaf::nio::FloatBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the float array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1919).

**6.366.3.3** `virtual int decaf::nio::FloatBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1920).

**6.366.3.4** `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only float buffer which the caller then owns.



Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1920).

### 6.366.3.5 virtual FloatBuffer& decaf::nio::FloatBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns:

a reference to this **FloatBuffer** (p. 1925).

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1921).

### 6.366.3.6 virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & value) const [virtual]

### 6.366.3.7 virtual FloatBuffer\* decaf::nio::FloatBuffer::duplicate () [pure virtual]

Creates a new float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

a new float **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1921).

### 6.366.3.8 virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & value) const [virtual]

### 6.366.3.9 FloatBuffer& decaf::nio::FloatBuffer::get (float \* buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if  $\text{length} > \text{remaining}()$  (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies length floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

**Parameters:**

*buffer* The pointer to an allocated buffer to fill.  
*size* The size of the passed in buffer.  
*offset* The position in the buffer to start filling.  
*length* The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length floats remaining in this buffer  
*NullPointerException* if the passed buffer is null.  
*IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

**6.366.3.10 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > *buffer*)  
 throw ( BufferUnderflowException )**

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length floats remaining in this buffer

**6.366.3.11 virtual float decaf::nio::FloatBuffer::get (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]**

Absolute get method. Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the float is to be read

**Returns:**

the float that is located at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1921).

### 6.366.3.12 virtual float decaf::nio::FloatBuffer::get () throw ( BufferUnderflowException ) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the float at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1922).

### 6.366.3.13 virtual bool decaf::nio::FloatBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible float array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1922).

### 6.366.3.14 virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & value) const [virtual]

### 6.366.3.15 virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & value) const [virtual]

### 6.366.3.16 virtual FloatBuffer& decaf::nio::FloatBuffer::put (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes the given floats into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The floats to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written, or index is negative.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1922).

**6.366.3.17** `virtual FloatBuffer& decaf::nio::FloatBuffer::put (float value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The floats value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1923).

**6.366.3.18** `FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & buffer) throw ( BufferOverflowException, ReadOnlyBufferException )`

This method transfers the entire content of the given source floats array into this buffer. This is the same as calling `put( &buffer[0], 0, buffer.size()`.

**Parameters:**

*buffer* The buffer whose contents are copied to this **FloatBuffer** (p. 1925)

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

**6.366.3.19** `FloatBuffer& decaf::nio::FloatBuffer::put (const float * buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

This method transfers floats into this buffer from the given source array. If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no floats are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters:**

- buffer* The array from which floats are to be read.
- size* The size of the passed in buffer.
- offset* The offset within the array of the first float to be read.
- length* The number of floats to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

- BufferOverflowException** (p. 954) if there is insufficient space in this buffer
- ReadOnlyBufferException** (p. 3169) if this buffer is read-only
- NullPointerException** if the passed buffer is null.
- IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

### 6.366.3.20 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the floats remaining in the given source buffer into this buffer. If there are more floats remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 933), then no floats are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

**Parameters:**

- src* The buffer to take floats from an place in this one.

**Returns:**

a reference to this buffer.

**Exceptions:**

- BufferOverflowException** (p. 954) if there is insufficient space in this buffer for the remaining floats in the source buffer
- IllegalArgumentException** if the source buffer is this buffer.
- ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

### 6.366.3.21 `virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const` [pure virtual]

Creates a new **FloatBuffer** (p. 1925) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

the newly create **FloatBuffer** (p. 1925) which the caller owns.

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1923).

### 6.366.3.22 `virtual std::string decaf::nio::FloatBuffer::toString () const` [virtual]

#### Returns:

a `std::string` describing this object

### 6.366.3.23 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & buffer)` [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1925). The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters:

*buffer* - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

#### Returns:

a new **FloatBuffer** (p. 1925) that is backed by *buffer*, caller owns.

### 6.366.3.24 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **FloatBuffer** (p. 1925). The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

- array* The array that will back the new buffer.
- size* The size of the array that was passed in.
- offset* The offset of the subarray to be used.
- length* The length of the subarray to be used.

**Returns:**

a new **FloatBuffer** (p. 1925) that is backed by buffer, caller owns.

**Exceptions:**

- NullPointerException* if the array pointer is NULL.
- IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

## 6.367 decaf::io::Flushable Class Reference

A **Flushable** (p. 1936) is a destination of data that can be flushed.

#include <src/main/decaf/io/Flushable.h> Inheritance diagram for decaf::io::Flushable:

### Public Member Functions

- virtual **~Flushable** ()
- virtual void **flush** ()=0 throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

### 6.367.1 Detailed Description

A **Flushable** (p. 1936) is a destination of data that can be flushed. The flush method is invoked to write any buffered output to the underlying stream.

Since:

1.0

### 6.367.2 Constructor & Destructor Documentation

**6.367.2.1** virtual decaf::io::Flushable::~~Flushable () [inline, virtual]

### 6.367.3 Member Function Documentation

**6.367.3.1** virtual void decaf::io::Flushable::flush () throw ( decaf::io::IOException )  
[pure virtual]

Flushes this stream by writing any buffered output to the underlying stream.

Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3580), **decaf::internal::io::StandardOutputStream** (p. 3585), **decaf::io::BufferedOutputStream** (p. 941), **decaf::io::FilterOutputStream** (p. 1902), **decaf::io::OutputStream** (p. 2910), and **decaf::io::OutputStreamWriter** (p. 2916).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Flushable.h**



## 6.368 activemq::commands::FlushCommand Class Reference

#include <src/main/activemq/commands/FlushCommand.h> Inheritance diagram for activemq::commands::FlushCommand:

### Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **FlushCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_FLUSHCOMMAND** = 15

## 6.368.1 Constructor & Destructor Documentation

**6.368.1.1** `activemq::commands::FlushCommand::FlushCommand ()`

**6.368.1.2** `virtual activemq::commands::FlushCommand::~~FlushCommand ()`  
[virtual]

## 6.368.2 Member Function Documentation

**6.368.2.1** `virtual FlushCommand* activemq::commands::FlushCommand::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.368.2.2** `virtual void activemq::commands::FlushCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.368.2.3** `virtual bool activemq::commands::FlushCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.368.2.4** `virtual unsigned char activemq::commands::FlushCommand::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

**6.368.2.5** `virtual std::string activemq::commands::FlushCommand::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.368.2.6** `virtual Pointer<Command> activemq::commands::FlushCommand::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.368.3 Field Documentation

**6.368.3.1** `const unsigned char activemq::commands::FlushCommand::ID _ - FLUSHCOMMAND = 15 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/FlushCommand.h`

## 6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1940).

#include <src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.369.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1940).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.369.2 Constructor & Destructor Documentation

**6.369.2.1** `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.369.2.2** `virtual activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.369.3 Member Function Documentation

**6.369.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.369.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.369.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.369.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.369.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.369.3.6** virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.369.3.7** virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h

## 6.370 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1944).

#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.370.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1944).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.370.2 Constructor & Destructor Documentation

**6.370.2.1** `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.370.2.2** `virtual activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.370.3 Member Function Documentation

**6.370.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.370.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.370.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.370.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.370.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.370.3.6** virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.370.3.7** virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h

## 6.371 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1948).

#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.371.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1948).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.371.2 Constructor & Destructor Documentation

**6.371.2.1** `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.371.2.2** `virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.371.3 Member Function Documentation

**6.371.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.371.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.371.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.371.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.371.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.371.3.6** virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.371.3.7** virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h

## 6.372 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1952).

#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.372.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1952).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.372.2 Constructor & Destructor Documentation

**6.372.2.1** `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.372.2.2** `virtual activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.372.3 Member Function Documentation

**6.372.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.372.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.372.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.372.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.372.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.372.3.6** virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.372.3.7** virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h

## 6.373 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1956).

#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.373.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **FlushCommandMarshaller** (p.1956).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.373.2 Constructor & Destructor Documentation

**6.373.2.1** `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.373.2.2** `virtual activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.373.3 Member Function Documentation

**6.373.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.373.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.373.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.373.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.373.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.373.3.6** virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.373.3.7** virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**

## 6.374 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1960).

#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.374.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1960).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.374.2 Constructor & Destructor Documentation

**6.374.2.1** `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

**6.374.2.2** `virtual activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

## 6.374.3 Member Function Documentation

**6.374.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.374.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.374.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.374.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.374.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.374.3.6** virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.374.3.7** virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h

## 6.375 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1964) provides support for formatting LogRecords.

#include <src/main/decaf/util/logging/Formatter.h> Inheritance diagram for decaf::util::logging::Formatter:

### Public Member Functions

- virtual **~Formatter** ()
- virtual std::string **format** (const **LogRecord** &record) const =0  
*Format the given log record and return the formatted string.*
- virtual std::string **formatMessage** (const **LogRecord** &record) const  
*Format the message string from a log record.*
- virtual std::string **getHead** (const **Handler** \*handler DECAF\_UNUSED)  
*Return the header string for a set of formatted records.*
- virtual std::string **getTail** (const **Handler** \*handler DECAF\_UNUSED)  
*Return the tail string for a set of formatted records.*

### 6.375.1 Detailed Description

A **Formatter** (p. 1964) provides support for formatting LogRecords. Typically each **logging** (p. 175) **Handler** (p. 1978) will have a **Formatter** (p. 1964) associated with it. The **Formatter** (p. 1964) takes a **LogRecord** (p. 2413) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p. 4060)) need to wrap head and tail strings around a set of formatted records. The **getHeader** and **getTail** methods can be used to obtain these strings.

### 6.375.2 Constructor & Destructor Documentation

**6.375.2.1** virtual decaf::util::logging::Formatter::~~Formatter () [inline, virtual]

### 6.375.3 Member Function Documentation

**6.375.3.1** virtual std::string decaf::util::logging::Formatter::format (const **LogRecord** & *record*) const [pure virtual]

Format the given log record and return the formatted string.

**Parameters:**

*record* The Log Record to Format

**Returns:**

the formatted record.

Implemented in `decaf::util::logging::SimpleFormatter` (p. 3500), and `decaf::util::logging::XMLFormatter` (p. 4060).

### 6.375.3.2 `virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & record) const` [virtual]

Format the message string from a log record.

#### Parameters:

*record* The Log Record to Format

#### Returns:

the formatted message

### 6.375.3.3 `virtual std::string decaf::util::logging::Formatter::getHead (const Handler *handler DECAF_UNUSED)` [inline, virtual]

Return the header string for a set of formatted records. In the default implementation this method should return empty string.

#### Parameters:

*handler* The target handler, can be NULL.

#### Returns:

the head string.

### 6.375.3.4 `virtual std::string decaf::util::logging::Formatter::getTail (const Handler *handler DECAF_UNUSED)` [inline, virtual]

Return the tail string for a set of formatted records. In the default implementation this method should return empty string

#### Parameters:

*handler* the target handler, can be null

#### Returns:

the tail string

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Formatter.h`

## 6.376 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p.1966) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

### Public Member Functions

- virtual **~Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0  
*Attempts to cancel execution of this task.*
- bool **isCancelled** () const =0  
*Returns true if this task was canceled before it completed normally.*
- bool **isDone** () const =0  
*Returns true if this task completed.*
- V **get** ()=0 throw ( CancellationException, InterruptedException, ExecutionException )  
*Waits if necessary for the computation to complete, and then retrieves its result.*
- V **get** (long long timeout, **TimeUnit** unit)=0 throw ( InterruptedException, ExecutionException, TimeoutException )  
*Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.*

### 6.376.1 Detailed Description

```
template<typename V> class decaf::util::concurrent::Future< V >
```

A **Future** (p.1966) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method **get** when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the **cancel** method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p.1966) for the sake of cancellability but not provide a usable result, you can declare types of the form **Future<void\*>** and return null as a result of the underlying task.

## 6.376.2 Constructor & Destructor Documentation

**6.376.2.1** `template<typename V > virtual decaf::util::concurrent::Future< V >::~~Future () [inline, virtual]`

## 6.376.3 Member Function Documentation

**6.376.3.1** `template<typename V > bool decaf::util::concurrent::Future< V >::cancel (bool mayInterruptIfRunning) [pure virtual]`

Attempts to cancel execution of this task. This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when cancel is called, this task should never run. If the task has already started, then the *mayInterruptIfRunning* parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to **isDone()** (p. 1968) will always return true. Subsequent calls to **isCancelled()** (p. 1968) will always return true if this method returned true.

### Parameters:

*mayInterruptIfRunning* - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.

### Returns:

false if the task could not be canceled, typically because it has already completed normally;  
true otherwise

**6.376.3.2** `template<typename V > V decaf::util::concurrent::Future< V >::get (long long timeout, TimeUnit unit) throw ( InterruptedException, ExecutionException, TimeoutException ) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

### Parameters:

*timeout* - the maximum time to wait

*unit* - the time unit of the timeout argument

### Returns:

the computed result

### Exceptions:

*CancellationException* (p. 1083) - if the computation was canceled

*ExecutionException* (p. 1866) - if the computation threw an exception

*InterruptedException* - if the current thread was interrupted while waiting

*TimeoutException* (p. 3787) - if the wait timed out

**6.376.3.3** `template<typename V > V decaf::util::concurrent::Future< V >::get () throw ( CancellationException, InterruptedException, ExecutionException ) [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

**Returns:**

the computed result.

**Exceptions:**

*CancellationException* (p. 1083) - if the computation was canceled

*ExecutionException* (p. 1866) - if the computation threw an exception

*InterruptedException* - if the current thread was interrupted while waiting

**6.376.3.4** `template<typename V > bool decaf::util::concurrent::Future< V >::isCancelled () const [pure virtual]`

Returns true if this task was canceled before it completed normally.

**Returns:**

true if this task was canceled before it completed

**6.376.3.5** `template<typename V > bool decaf::util::concurrent::Future< V >::isDone () const [pure virtual]`

Returns true if this task completed. Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

**Returns:**

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Future.h`



## 6.377 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

### Public Member Functions

- **FutureResponse** ()
- virtual **~FutureResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** () const  
*Getters for the response property.*
- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const  
*Getters for the response property.*
- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)  
*Setter for the response property.*

### 6.377.1 Detailed Description

A container that holds a response object. Callers of the `getResponse` method will block until a response has been receive unless they call the `getRepsonse` that takes a timeout.

### 6.377.2 Constructor & Destructor Documentation

**6.377.2.1** `activemq::transport::correlator::FutureResponse::FutureResponse ()`  
[inline]

**6.377.2.2** `virtual`  
`activemq::transport::correlator::FutureResponse::~~FutureResponse ()`  
[inline, virtual]

### 6.377.3 Member Function Documentation

**6.377.3.1** `virtual` `Pointer`<`Response`>& `activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` [inline, virtual]

**6.377.3.2** `virtual` const `Pointer`<`Response`>& `activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` const [inline, virtual]

Getters for the response property. Timed Wait.

**Parameters:**

*timeout* - time to wait in milliseconds

**Returns:**

the response object for the request

**6.377.3.3** `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () [inline, virtual]`

**6.377.3.4** `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () const [inline, virtual]`

Getters for the response property. Infinite Wait.

**Returns:**

the response object for the request

**6.377.3.5** `virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & response) [inline, virtual]`

Setter for the response property.

**Parameters:**

*response* the response object for the request.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

## 6.378 decaf::security::GeneralSecurityException Class Reference

#include <src/main/decaf/security/GeneralSecurityException.h> Inheritance diagram for decaf::security::GeneralSecurityException:

### Public Member Functions

- **GeneralSecurityException** () throw ()  
*Default Constructor.*
- **GeneralSecurityException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **GeneralSecurityException** (const GeneralSecurityException &ex) throw ()  
*Copy Constructor.*
- **GeneralSecurityException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **GeneralSecurityException** (const std::exception \*cause) throw ()  
*Constructor.*
- **GeneralSecurityException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **GeneralSecurityException** \* clone () const  
*Clones this exception.*
- virtual ~**GeneralSecurityException** () throw ()

### 6.378.1 Constructor & Destructor Documentation

#### 6.378.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException () throw () [inline]

Default Constructor.

#### 6.378.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.378.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.378.1.4 decaf::security::GeneralSecurityException::GeneralSecurityException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.378.1.5 decaf::security::GeneralSecurityException::GeneralSecurityException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.378.1.6 decaf::security::GeneralSecurityException::GeneralSecurityException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.378.1.7** virtual  
 decaf::security::GeneralSecurityException::~~GeneralSecurityException ()  
 throw () [inline, virtual]

## 6.378.2 Member Function Documentation

**6.378.2.1** virtual GeneralSecurityException\* de-  
 caf::security::GeneralSecurityException::clone () const  
 [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A deep copy of this exception.

Reimplemented from **decaf::lang::Exception** (p. 1834).

Reimplemented in **decaf::security::cert::CertificateEncodingException**  
 (p. 1091), **decaf::security::cert::CertificateException** (p. 1093), **de-  
 caf::security::cert::CertificateExpiredException** (p. 1095), **de-  
 caf::security::cert::CertificateNotYetValidException** (p. 1097), **de-  
 caf::security::cert::CertificateParsingException** (p. 1099), **de-  
 caf::security::InvalidKeyException** (p. 2134), **decaf::security::KeyException**  
 (p. 2297), **decaf::security::KeyManagementException** (p. 2300),  
**decaf::security::NoSuchAlgorithmException** (p. 2825), **de-  
 caf::security::NoSuchProviderException** (p. 2831), and **de-  
 caf::security::SignatureException** (p. 3499).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**GeneralSecurityException.h**

## 6.379 decaf::internal::util::GenericResource< T > Class Template Reference

A Generic **Resource** (p. 3280) wraps some type and will delete it when the **Resource** (p. 3280) itself is deleted.

#include <src/main/decaf/internal/util/GenericResource.h> Inheritance diagram for decaf::internal::util::GenericResource< T >:

### Public Member Functions

- **GenericResource** (T \*value)
- virtual ~**GenericResource** ()
- T \* **getManaged** () const
- void **setManaged** (T \*value)

### 6.379.1 Detailed Description

```
template<typename T> class decaf::internal::util::GenericResource< T >
```

A Generic **Resource** (p. 3280) wraps some type and will delete it when the **Resource** (p. 3280) itself is deleted.

Since:

1.0

### 6.379.2 Constructor & Destructor Documentation

**6.379.2.1** `template<typename T> decaf::internal::util::GenericResource< T >::GenericResource (T * value) [inline, explicit]`

**6.379.2.2** `template<typename T> virtual decaf::internal::util::GenericResource< T >::~~GenericResource () [inline, virtual]`

### 6.379.3 Member Function Documentation

**6.379.3.1** `template<typename T> T* decaf::internal::util::GenericResource< T >::getManaged () const [inline]`

**6.379.3.2** `template<typename T> void decaf::internal::util::GenericResource< T >::setManaged (T * value) [inline]`

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**GenericResource.h**

## 6.380 gz\_header\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

### Data Fields

- int text
- uLong time
- int xflags
- int os
- Bytef \* extra
- uInt extra\_len
- uInt extra\_max
- Bytef \* name
- uInt name\_max
- Bytef \* comment
- uInt comm\_max
- int hcrc
- int done

### 6.380.1 Field Documentation

6.380.1.1 uInt gz\_header\_s::comm\_max

6.380.1.2 Bytef\* gz\_header\_s::comment

6.380.1.3 int gz\_header\_s::done

6.380.1.4 Bytef\* gz\_header\_s::extra

6.380.1.5 uInt gz\_header\_s::extra\_len

6.380.1.6 uInt gz\_header\_s::extra\_max

6.380.1.7 int gz\_header\_s::hcrc

6.380.1.8 Bytef\* gz\_header\_s::name

6.380.1.9 uInt gz\_header\_s::name\_max

6.380.1.10 int gz\_header\_s::os

6.380.1.11 int gz\_header\_s::text

6.380.1.12 uLong gz\_header\_s::time

6.380.1.13 int gz\_header\_s::xflags

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/zlib.h

## 6.381 gz\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/gzguts.h>
```

### Data Fields

- int **mode**
- int **fd**
- char \* **path**
- z\_off64\_t **pos**
- unsigned **size**
- unsigned **want**
- unsigned char \* **in**
- unsigned char \* **out**
- unsigned char \* **next**
- unsigned **have**
- int **eof**
- z\_off64\_t **start**
- z\_off64\_t **raw**
- int **how**
- int **direct**
- int **level**
- int **strategy**
- z\_off64\_t **skip**
- int **seek**
- int **err**
- char \* **msg**
- z\_stream **strm**



### 6.381.1 Field Documentation

- 6.381.1.1 `int gz_state::direct`
- 6.381.1.2 `int gz_state::eof`
- 6.381.1.3 `int gz_state::err`
- 6.381.1.4 `int gz_state::fd`
- 6.381.1.5 `unsigned gz_state::have`
- 6.381.1.6 `int gz_state::how`
- 6.381.1.7 `unsigned char* gz_state::in`
- 6.381.1.8 `int gz_state::level`
- 6.381.1.9 `int gz_state::mode`
- 6.381.1.10 `char* gz_state::msg`
- 6.381.1.11 `unsigned char* gz_state::next`
- 6.381.1.12 `unsigned char* gz_state::out`
- 6.381.1.13 `char* gz_state::path`
- 6.381.1.14 `z_off64_t gz_state::pos`
- 6.381.1.15 `z_off64_t gz_state::raw`
- 6.381.1.16 `int gz_state::seek`
- 6.381.1.17 `unsigned gz_state::size`
- 6.381.1.18 `z_off64_t gz_state::skip`
- 6.381.1.19 `z_off64_t gz_state::start`
- 6.381.1.20 `int gz_state::strategy`
- 6.381.1.21 `z_stream gz_state::strm`
- 6.381.1.22 `unsigned gz_state::want`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/gzguts.h`

## 6.382 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1978) object takes log messages from a **Logger** (p. 2386) and exports them.

#include <src/main/decaf/util/logging/Handler.h> Inheritance diagram for decaf::util::logging::Handler:

### Public Member Functions

- **Handler** ()
- virtual **~Handler** ()
- virtual void **flush** ()=0  
*Flush the Handler's output, clears any buffers.*
- virtual void **publish** (const **LogRecord** &record)=0  
*Publish the Log Record to this **Handler** (p. 1978).*
- virtual bool **isLoggable** (const **LogRecord** &record) const  
*Check if this **Handler** (p. 1978) would actually log a given **LogRecord** (p. 2413).*
- virtual void **setFilter** (**Filter** \*filter)  
*Sets the **Filter** (p. 1893) that this **Handler** (p. 1978) uses to filter Log Records.*
- virtual **Filter** \* **getFilter** ()  
*Gets the **Filter** (p. 1893) that this **Handler** (p. 1978) uses to filter Log Records.*
- virtual void **setLevel** (const **Level** &value)  
*Set (p. 3439) the log level specifying which message levels will be logged by this **Handler** (p. 1978).*
- virtual **Level** **getLevel** ()  
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1978).*
- virtual void **setFormatter** (**Formatter** \*formatter)  
*Sets the **Formatter** (p. 1964) used by this **Handler** (p. 1978).*
- virtual **Formatter** \* **getFormatter** ()  
*Gets the **Formatter** (p. 1964) used by this **Handler** (p. 1978).*
- virtual void **setErrorManager** (**ErrorManager** \*errorManager)  
*Sets the **Formatter** (p. 1964) used by this **Handler** (p. 1978).*
- virtual **ErrorManager** \* **getErrorManager** ()  
*Gets the **ErrorManager** (p. 1828) used by this **Handler** (p. 1978).*

## Protected Member Functions

- void **reportError** (const std::string &message, **decaf::lang::Exception** \*ex, int code)  
*Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1828).*

### 6.382.1 Detailed Description

A **Handler** (p.1978) object takes log messages from a **Logger** (p.2386) and exports them. It might for example, write them to a console or write them to a file, or send them to a network **logging** (p.175) service, or forward them to an OS log, or whatever.

A **Handler** (p.1978) can be disabled by doing a **setLevel(Level.OFF** (p.2336)) and can be re-enabled by doing a **setLevel** with an appropriate level.

**Handler** (p.1978) classes typically use **LogManager** (p.2406) properties to set default values for the Handler's **Filter** (p.1893), **Formatter** (p.1964), and **Level** (p.2332). See the specific documentation for each concrete **Handler** (p.1978) class.

### 6.382.2 Constructor & Destructor Documentation

**6.382.2.1** **decaf::util::logging::Handler::Handler** ()

**6.382.2.2** **virtual decaf::util::logging::Handler::~~Handler** () [virtual]

### 6.382.3 Member Function Documentation

**6.382.3.1** **virtual void decaf::util::logging::Handler::flush** () [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p.3651).

**6.382.3.2** **virtual ErrorManager\* decaf::util::logging::Handler::getErrorManager** ()  
[inline, virtual]

Gets the **ErrorManager** (p.1828) used by this **Handler** (p.1978).

#### Returns:

**ErrorManager** (p.1828) derived pointer or NULL.

**6.382.3.3** **virtual Filter\* decaf::util::logging::Handler::getFilter** () [inline, virtual]

Gets the **Filter** (p.1893) that this **Handler** (p.1978) uses to filter Log Records.

#### Returns:

**Filter** (p.1893) derived instance

#### 6.382.3.4 virtual Formatter\* decaf::util::logging::Handler::getFormatter () [inline, virtual]

Gets the **Formatter** (p. 1964) used by this **Handler** (p. 1978).

##### Returns:

**Filter** (p. 1893) derived instance

#### 6.382.3.5 virtual Level decaf::util::logging::Handler::getLevel () [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1978).

##### Returns:

**Level** (p. 2332) enumeration value

#### 6.382.3.6 virtual bool decaf::util::logging::Handler::isLoggable (const LogRecord & record) const [virtual]

Check if this **Handler** (p. 1978) would actually log a given **LogRecord** (p. 2413). This method checks if the **LogRecord** (p. 2413) has an appropriate **Level** (p. 2332) and whether it satisfies any **Filter** (p. 1893). It also may make other **Handler** (p. 1978) specific checks that might prevent a handler from **logging** (p. 175) the **LogRecord** (p. 2413).

##### Parameters:

*record* **LogRecord** (p. 2413) to check

Reimplemented in **decaf::util::logging::StreamHandler** (p. 3651).

#### 6.382.3.7 virtual void decaf::util::logging::Handler::publish (const LogRecord & record) [pure virtual]

Publish the Log Record to this **Handler** (p. 1978).

##### Parameters:

*record* The Log Record to Publish

Implemented in **decaf::util::logging::ConsoleHandler** (p. 1400), and **decaf::util::logging::StreamHandler** (p. 3651).

#### 6.382.3.8 void decaf::util::logging::Handler::reportError (const std::string & message, decaf::lang::Exception \* ex, int code) [protected]

Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1828).

##### Parameters:

*message* - a descriptive string (may be empty)

*ex* - an exception (may be NULL)

*code* (p. 1183) - an error **code** (p. 1183) defined in **ErrorManager** (p. 1828)

**6.382.3.9 virtual void decaf::util::logging::Handler::setErrorManager (ErrorManager \* *errorManager*) [virtual]**

Sets the **Formatter** (p.1964) used by this **Handler** (p.1978). The **ErrorManager**'s "error" method will be invoked if any errors occur while using this **Handler** (p.1978).

**Parameters:**

*errorManager* **ErrorManager** (p.1828) derived instance

**6.382.3.10 virtual void decaf::util::logging::Handler::setFilter (Filter \* *filter*) [inline, virtual]**

Sets the **Filter** (p.1893) that this **Handler** (p.1978) uses to filter Log Records. For each call of publish the **Handler** (p.1978) will call this **Filter** (p.1893) (if it is non-null) to check if the **LogRecord** (p.2413) should be published or discarded.

**Parameters:**

*filter* **Filter** (p.1893) derived instance

**6.382.3.11 virtual void decaf::util::logging::Handler::setFormatter (Formatter \* *formatter*) [virtual]**

Sets the **Formatter** (p.1964) used by this **Handler** (p.1978). Some **Handlers** may not use **Formatters**, in which case the **Formatter** (p.1964) will be remembered, but not used.

**Parameters:**

*formatter* **Filter** (p.1893) derived instance

**6.382.3.12 virtual void decaf::util::logging::Handler::setLevel (const Level & *value*) [inline, virtual]**

**Set** (p.3439) the log level specifying which message levels will be logged by this **Handler** (p.1978). The intention is to allow developers to turn on voluminous **logging** (p.175), but to limit the messages that are sent to certain **Handlers**.

**Parameters:**

*value* **Level** (p.2332) enumeration value

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

## 6.383 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

### Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)  
*Create a new HexParser.*
- virtual **~HexStringParser** ()
- long long **parse** (const std::string &hexString)  
*Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.*

### Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

### 6.383.1 Constructor & Destructor Documentation

#### 6.383.1.1 decaf::internal::util::HexStringParser::HexStringParser (int exponentWidth, int mantissaWidth)

Create a new HexParser.

##### Parameters:

*exponentWidth* - Width of the exponent for the type to parse  
*mantissaWidth* - Width of the mantissa for the type to parse

#### 6.383.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser () [inline, virtual]

### 6.383.2 Member Function Documentation

#### 6.383.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string &hexString)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

##### Parameters:

*hexString* - string to parse

##### Returns:

the bits parsed from the string

**6.383.2.2** static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

**6.383.2.3** static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

## 6.384 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p.1984) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

### Public Member Functions

- **HexTable** ()
- virtual **~HexTable** ()
- virtual const std::string & **operator[]** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.*

- virtual const std::string & **operator[]** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **size** () const

*Returns the max size of this Table.*

### 6.384.1 Detailed Description

The **HexTable** (p.1984) class maps hexadecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

### 6.384.2 Constructor & Destructor Documentation

**6.384.2.1** activemq::wireformat::openwire::utils::HexTable::HexTable ()

**6.384.2.2** virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ()  
[inline, virtual]

### 6.384.3 Member Function Documentation

**6.384.3.1** virtual const std::string&  
activemq::wireformat::openwire::utils::HexTable::operator[] (std::size\_t  
*index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
) [virtual]

**6.384.3.2** virtual const std::string&  
activemq::wireformat::openwire::utils::HexTable::operator[] (std::size\_t  
*index*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.



**Parameters:**

*index* The index of the value in the table to fetch.

**Returns:**

string containing the hex value if the index

**Exceptions:**

*IndexOutOfBoundsException*

**6.384.3.3** `virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size  
() const [inline, virtual]`

Returns the max size of this Table.

**Returns:**

an integer size value

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/HexTable.h`

## 6.385 decaf::net::HttpRetryException Class Reference

#include <src/main/decaf/net/HttpRetryException.h> Inheritance diagram for decaf::net::HttpRetryException:

### Public Member Functions

- **HttpRetryException** () throw ()  
*Default Constructor.*
- **HttpRetryException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()  
*Copy Constructor.*
- **HttpRetryException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **HttpRetryException** (const std::exception \*cause) throw ()  
*Constructor.*
- **HttpRetryException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **HttpRetryException** \* **clone** () const  
*Clones this exception.*
- virtual ~**HttpRetryException** () throw ()

### 6.385.1 Constructor & Destructor Documentation

#### 6.385.1.1 decaf::net::HttpRetryException::HttpRetryException () throw () [inline]

Default Constructor.

#### 6.385.1.2 decaf::net::HttpRetryException::HttpRetryException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.385.1.3 decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.385.1.4 decaf::net::HttpRetryException::HttpRetryException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.385.1.5 decaf::net::HttpRetryException::HttpRetryException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.385.1.6 decaf::net::HttpRetryException::HttpRetryException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.385.1.7** `virtual decaf::net::HttpRetryException::~~HttpRetryException () throw  
() [inline, virtual]`

## **6.385.2 Member Function Documentation**

**6.385.2.1** `virtual HttpRetryException* decaf::net::HttpRetryException::clone ()  
const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/HttpRetryException.h`

## 6.386 activemq::util::IdGenerator Class Reference

```
#include <src/main/activemq/util/IdGenerator.h>
```

### Data Structures

- class **StaticData**

### Public Member Functions

- **IdGenerator** ()
- **IdGenerator** (const std::string &prefix)
- virtual **~IdGenerator** ()
- std::string **generateId** () const

### Static Public Member Functions

- static std::string **getHostname** ()  
*Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.*
- static std::string **getSeedFromId** (const std::string &id)  
*Gets the seed value from a Generated Id, the count portion is removed.*
- static long long **getSequenceFromId** (const std::string &id)  
*Gets the count value from a Generated Id, the seed portion is removed.*
- static int **compare** (const std::string &id1, const std::string &id2)  
*Compares two generated id values.*

### 6.386.1 Constructor & Destructor Documentation

**6.386.1.1** **activemq::util::IdGenerator::IdGenerator** ()

**6.386.1.2** **activemq::util::IdGenerator::IdGenerator** (const std::string & *prefix*)

**6.386.1.3** **virtual activemq::util::IdGenerator::~~IdGenerator** () [virtual]

### 6.386.2 Member Function Documentation

**6.386.2.1** **static int activemq::util::IdGenerator::compare** (const std::string & *id1*, const std::string & *id2*) [static]

Compares two generated id values.

#### Parameters:

*id1* The first id to compare, or left hand side.

*id2* The second id to compare, or right hand side.

**Returns:**

zero if ids are equal or positive if  $id1 > id2$ ...

**6.386.2.2** `std::string activemq::util::IdGenerator::generateId () const`**Returns:**

a newly generated unique id.

**6.386.2.3** `static std::string activemq::util::IdGenerator::getHostname () [static]`

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

**Returns:**

the previously retrieved host name.

**6.386.2.4** `static std::string activemq::util::IdGenerator::getSeedFromId (const std::string & id) [static]`

Gets the seed value from a Generated Id, the count portion is removed.

**Returns:**

the seed portion of the Id, minus the count value.

**6.386.2.5** `static long long activemq::util::IdGenerator::getSequenceFromId (const std::string & id) [static]`

Gets the count value from a Generated Id, the seed portion is removed.

**Returns:**

the sequence count portion of the id, minus the seed value.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/IdGenerator.h`

## 6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>Inheritance diagram for decaf::lang::exceptions::IllegalArgumentException:

### Public Member Functions

- **IllegalArgumentException** () throw ()  
*Default Constructor.*
- **IllegalArgumentException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()  
*Copy Constructor.*
- **IllegalArgumentException** (const std::exception \*cause) throw ()  
*Constructor.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **IllegalArgumentException** \* **clone** () const  
*Clones this exception.*
- virtual ~**IllegalArgumentException** () throw ()

### 6.387.1 Constructor & Destructor Documentation

#### 6.387.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException () throw () [inline]

Default Constructor.

#### 6.387.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

### 6.387.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

### 6.387.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.387.1.5 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.387.1.6 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message



**6.387.1.7** virtual  
decaf::lang::exceptions::IllegalArgumentException::~~IllegalArgumentException  
( ) throw ( ) [inline, virtual]

## 6.387.2 Member Function Documentation

**6.387.2.1** virtual IllegalArgumentException\* de-  
cafe::lang::exceptions::IllegalArgumentException::clone ( )  
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalArgumentException.h**

## 6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

### Public Member Functions

- **IllegalMonitorStateException** () throw ()  
*Default Constructor.*
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalMonitorStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalMonitorStateException** () throw ()

### 6.388.1 Constructor & Destructor Documentation

#### 6.388.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException () throw () [inline]

Default Constructor.

#### 6.388.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.388.1.3 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const IllegalMonitorStateException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.388.1.4 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.388.1.5 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.388.1.6 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.388.1.7**    **virtual**  
          `decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException`  
          `() throw ()`    `[inline, virtual]`

## **6.388.2    Member Function Documentation**

**6.388.2.1**    **virtual** `IllegalMonitorStateException*` `de-`  
          `caf::lang::exceptions::IllegalMonitorStateException::clone`  
          `() const`    `[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalMonitorStateException.h`

## 6.389 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

#include <src/main/cms/IllegalStateException.h> Inheritance diagram for cms::IllegalStateException:

### Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception \*cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

### 6.389.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example, this exception must be thrown if **Session.commit** (p. 3369) is called on a non-transacted session.

Since:

1.3

### 6.389.2 Constructor & Destructor Documentation

**6.389.2.1** cms::IllegalStateException::IllegalStateException () throw ()

**6.389.2.2** cms::IllegalStateException::IllegalStateException (const **IllegalStateException** & ex) throw ()

**6.389.2.3** cms::IllegalStateException::IllegalStateException (const std::string & message, const std::exception \* cause) throw ()

**6.389.2.4** cms::IllegalStateException::IllegalStateException (const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()

**6.389.2.5** virtual cms::IllegalStateException::~~IllegalStateException () throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**IllegalStateException.h**

## 6.390 decaf::lang::exceptions::IllegalStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalStateException:

### Public Member Functions

- **IllegalStateException** () throw ()  
*Default Constructor.*
- **IllegalStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalStateException** () throw ()

### 6.390.1 Constructor & Destructor Documentation

#### 6.390.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException () throw () [inline]

Default Constructor.

#### 6.390.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.390.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException**  
(const IllegalStateException & *ex*) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.390.1.4 decaf::lang::exceptions::IllegalStateException::IllegalStateException**  
(const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.390.1.5 decaf::lang::exceptions::IllegalStateException::IllegalStateException**  
(const char \* *file*, const int *lineNumber*, const std::exception \* *cause*,  
const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.390.1.6 decaf::lang::exceptions::IllegalStateException::IllegalStateException**  
(const std::exception \* *cause*) throw () [inline]

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.390.1.7**   **virtual**  
**decaf::lang::exceptions::IllegalStateException::~~IllegalStateException ()**  
**throw ()**   [inline, virtual]

## **6.390.2   Member Function Documentation**

**6.390.2.1**   **virtual** **IllegalStateException\*** **de-**  
**caf::lang::exceptions::IllegalStateException::clone ()** **const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 2137).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`



## 6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalThreadStateException:

### Public Member Functions

- **IllegalThreadStateException** () throw ()  
*Default Constructor.*
- **IllegalThreadStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalThreadStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalThreadStateException** () throw ()

### 6.391.1 Constructor & Destructor Documentation

#### 6.391.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException () throw () [inline]

Default Constructor.

#### 6.391.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

### 6.391.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const IllegalThreadStateException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

### 6.391.1.4 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.391.1.5 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.391.1.6 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.391.1.7** virtual  
decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException  
( ) throw ( ) [inline, virtual]

## 6.391.2 Member Function Documentation

**6.391.2.1** virtual IllegalThreadStateException\* de-  
caf::lang::exceptions::IllegalThreadStateException::clone ( )  
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalThreadStateException.h**

## 6.392 activemq::transport::inactivity::InactivityMonitor

### Class Reference

#include <src/main/activemq/transport/inactivity/InactivityMonitor.h> Inheritance diagram for activemq::transport::inactivity::InactivityMonitor:

#### Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual ~**InactivityMonitor** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

#### Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

## 6.392.1 Constructor & Destructor Documentation

- 6.392.1.1** `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.392.1.2** `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.392.1.3** `virtual  
activemq::transport::inactivity::InactivityMonitor::~~InactivityMonitor ()  
[virtual]`

## 6.392.2 Member Function Documentation

- 6.392.2.1** `virtual void activemq::transport::inactivity::InactivityMonitor::close ()  
throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### Exceptions:

*IOException* if an error occurs while closing the **Transport** (p. 3883).

Reimplemented from **activemq::transport::TransportFilter** (p. 3893).

- 6.392.2.2** `long long ac-  
tivemq::transport::inactivity::InactivityMonitor::getInitialDelayTime ()  
const`
- 6.392.2.3** `long long ac-  
tivemq::transport::inactivity::InactivityMonitor::getReadCheckTime ()  
const`
- 6.392.2.4** `long long ac-  
tivemq::transport::inactivity::InactivityMonitor::getWriteCheckTime ()  
const`
- 6.392.2.5** `bool ac-  
tivemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired  
() const`
- 6.392.2.6** `virtual void ac-  
tivemq::transport::inactivity::InactivityMonitor::onCommand (const  
Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

### Parameters:

*command* - the received command object.

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

**6.392.2.7** `virtual void activemq::transport::inactivity::InactivityMonitor::oneway  
(const Pointer< Command > & command) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

**Parameters:**

*command* the command to be sent.

**Exceptions:**

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

**6.392.2.8** `virtual void ac-  
tivemq::transport::inactivity::InactivityMonitor::onException (const  
decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command **transport** (p. 97).

**Parameters:**

*ex* The exception to handle.

Reimplemented from **activemq::transport::TransportFilter** (p. 3896).

- 6.392.2.9** void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime (long long *value*) const
- 6.392.2.10** void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired (bool *value*)
- 6.392.2.11** void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime (long long *value*)
- 6.392.2.12** void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime (long long *value*)

### 6.392.3 Friends And Related Function Documentation

- 6.392.3.1** friend class AsyncSignalReadErrorTask [friend]
- 6.392.3.2** friend class AsyncWriteTask [friend]
- 6.392.3.3** friend class ReadChecker [friend]
- 6.392.3.4** friend class WriteChecker [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**InactivityMonitor.h**

## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

### Public Member Functions

- **IndexOutOfBoundsException** () throw ()  
*Default Constructor.*
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()  
*Copy Constructor.*
- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IndexOutOfBoundsException** \* clone () const  
*Clones this exception.*
- virtual ~**IndexOutOfBoundsException** () throw ()

### 6.393.1 Constructor & Destructor Documentation

#### 6.393.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw () [inline]

Default Constructor.

#### 6.393.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.



**6.393.1.3 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.393.1.4 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.393.1.5 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.393.1.6 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.393.1.7**   **virtual**  
**decaf::lang::exceptions::IndexOutOfBoundsException::~~IndexOutOfBoundsException**  
**() throw ()**   [inline, virtual]

## **6.393.2**   **Member Function Documentation**

**6.393.2.1**   **virtual IndexOutOfBoundsException\* de-**  
**caf::lang::exceptions::IndexOutOfBoundsException::clone**  
**() const**   [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IndexOutOfBoundsException.h**

## 6.394 decaf::net::Inet4Address Class Reference

#include <src/main/decaf/net/Inet4Address.h> Inheritance diagram for decaf::net::Inet4Address:

### Public Member Functions

- virtual `~Inet4Address ()`
- virtual bool `isAnyLocalAddress ()` const  
*Check if this **InetAddress** (p. 2016) is a valid wildcard address.*
- virtual bool `isLoopbackAddress ()` const  
*Check if this **InetAddress** (p. 2016) is a valid loopback address.*
- virtual bool `isMulticastAddress ()` const  
*Check if this **InetAddress** (p. 2016) is a valid Multicast address.*
- virtual bool `isLinkLocalAddress ()` const  
*Check if this **InetAddress** (p. 2016) is a valid link local address.*
- virtual bool `isSiteLocalAddress ()` const  
*Check if this **InetAddress** (p. 2016) is a valid site local address.*
- virtual bool `isMCGlobal ()` const  
*Check if this **InetAddress** (p. 2016) is Multicast and has Global scope.*
- virtual bool `isMCNodeLocal ()` const  
*Check if this **InetAddress** (p. 2016) is Multicast and has Node Local scope.*
- virtual bool `isMCLinkLocal ()` const  
*Check if this **InetAddress** (p. 2016) is Multicast and has Link Local scope.*
- virtual bool `isMCSiteLocal ()` const  
*Check if this **InetAddress** (p. 2016) is Multicast and has Site Local scope.*
- virtual bool `isMCOrgLocal ()` const  
*Check if this **InetAddress** (p. 2016) is Multicast and has Organization scope.*

### Protected Member Functions

- `Inet4Address ()`
- `Inet4Address (const unsigned char *ipAddress, int numBytes)`
- `Inet4Address (const std::string &hostname, const unsigned char *ipAddress, int numBytes)`

## Friends

- class `InetAddress`

### 6.394.1 Constructor & Destructor Documentation

- 6.394.1.1** `decaf::net::Inet4Address::Inet4Address ()` [protected]
- 6.394.1.2** `decaf::net::Inet4Address::Inet4Address (const unsigned char * ipAddress, int numBytes)` [protected]
- 6.394.1.3** `decaf::net::Inet4Address::Inet4Address (const std::string & hostname, const unsigned char * ipAddress, int numBytes)` [protected]
- 6.394.1.4** `virtual decaf::net::Inet4Address::~~Inet4Address ()` [virtual]

### 6.394.2 Member Function Documentation

- 6.394.2.1** `virtual bool decaf::net::Inet4Address::isAnyLocalAddress () const` [virtual]

Check if this `InetAddress` (p. 2016) is a valid wildcard address.

#### Returns:

true if the address is a wildcard address.

Reimplemented from `decaf::net::InetAddress` (p. 2020).

- 6.394.2.2** `virtual bool decaf::net::Inet4Address::isLinkLocalAddress () const` [virtual]

Check if this `InetAddress` (p. 2016) is a valid link local address.

#### Returns:

true if the address is a link local address.

Reimplemented from `decaf::net::InetAddress` (p. 2020).

- 6.394.2.3** `virtual bool decaf::net::Inet4Address::isLoopbackAddress () const` [virtual]

Check if this `InetAddress` (p. 2016) is a valid loopback address.

#### Returns:

true if the address is a loopback address.

Reimplemented from `decaf::net::InetAddress` (p. 2020).

**6.394.2.4 virtual bool decaf::net::Inet4Address::isMCGlobal () const [virtual]**

Check if this **InetAddress** (p. 2016) is Multicast and has Global scope.

**Returns:**

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 2021).

**6.394.2.5 virtual bool decaf::net::Inet4Address::isMCLinkLocal () const [virtual]**

Check if this **InetAddress** (p. 2016) is Multicast and has Link Local scope.

**Returns:**

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2021).

**6.394.2.6 virtual bool decaf::net::Inet4Address::isMCNodeLocal () const [virtual]**

Check if this **InetAddress** (p. 2016) is Multicast and has Node Local scope.

**Returns:**

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2021).

**6.394.2.7 virtual bool decaf::net::Inet4Address::isMCOrgLocal () const [virtual]**

Check if this **InetAddress** (p. 2016) is Multicast and has Organization scope.

**Returns:**

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 2021).

**6.394.2.8 virtual bool decaf::net::Inet4Address::isMCSiteLocal () const [virtual]**

Check if this **InetAddress** (p. 2016) is Multicast and has Site Local scope.

**Returns:**

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2021).

**6.394.2.9**    **virtual bool decaf::net::Inet4Address::isMulticastAddress () const**  
                  [virtual]

Check if this **InetAddress** (p. 2016) is a valid Multicast address.

**Returns:**

    true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 2022).

**6.394.2.10**    **virtual bool decaf::net::Inet4Address::isSiteLocalAddress () const**  
                  [virtual]

Check if this **InetAddress** (p. 2016) is a valid site local address.

**Returns:**

    true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 2022).

## **6.394.3    Friends And Related Function Documentation**

**6.394.3.1**    **friend class InetAddress**    [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet4Address.h`

## 6.395 decaf::net::Inet6Address Class Reference

#include <src/main/decaf/net/Inet6Address.h> Inheritance diagram for decaf::net::Inet6Address:

### Public Member Functions

- virtual ~Inet6Address ()

### Protected Member Functions

- Inet6Address ()
- Inet6Address (const unsigned char \*ipAddress, int numBytes)
- Inet6Address (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

### Friends

- class InetAddress

### 6.395.1 Constructor & Destructor Documentation

6.395.1.1 decaf::net::Inet6Address::Inet6Address () [protected]

6.395.1.2 decaf::net::Inet6Address::Inet6Address (const unsigned char \* *ipAddress*, int *numBytes*) [protected]

6.395.1.3 decaf::net::Inet6Address::Inet6Address (const std::string & *hostname*, const unsigned char \* *ipAddress*, int *numBytes*) [protected]

6.395.1.4 virtual decaf::net::Inet6Address::~~Inet6Address () [virtual]

### 6.395.2 Friends And Related Function Documentation

6.395.2.1 friend class InetAddress [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/net/Inet6Address.h

## 6.396 decaf::net::InetAddress Class Reference

Represents an IP address.

#include <src/main/decaf/net/InetAddress.h> Inheritance diagram for decaf::net::InetAddress:

### Public Member Functions

- virtual `~InetAddress ()`
- virtual `decaf::lang::ArrayPointer< unsigned char > getAddress () const`  
*Returns the Raw IP address in Network byte order.*
- virtual `std::string getHostAddress () const`  
*Returns a textual representation of the IP Address.*
- virtual `std::string getHostName () const`  
*Get the host name associated with this **InetAddress** (p. 2016) instance.*
- virtual `std::string toString () const`  
*Returns a string representation of the **InetAddress** (p. 2016) in the form 'hostname / ipaddress'.*
- virtual `bool isAnyLocalAddress () const`  
*Check if this **InetAddress** (p. 2016) is a valid wildcard address.*
- virtual `bool isLoopbackAddress () const`  
*Check if this **InetAddress** (p. 2016) is a valid loopback address.*
- virtual `bool isMulticastAddress () const`  
*Check if this **InetAddress** (p. 2016) is a valid Multicast address.*
- virtual `bool isLinkLocalAddress () const`  
*Check if this **InetAddress** (p. 2016) is a valid link local address.*
- virtual `bool isSiteLocalAddress () const`  
*Check if this **InetAddress** (p. 2016) is a valid site local address.*
- virtual `bool isMCGlobal () const`  
*Check if this **InetAddress** (p. 2016) is Multicast and has Global scope.*
- virtual `bool isMCNodeLocal () const`  
*Check if this **InetAddress** (p. 2016) is Multicast and has Node Local scope.*
- virtual `bool isMCLinkLocal () const`  
*Check if this **InetAddress** (p. 2016) is Multicast and has Link Local scope.*
- virtual `bool isMCSiteLocal () const`



Check if this **InetAddress** (p. 2016) is Multicast and has Site Local scope.

- virtual bool **isMCOrgLocal** () const

Check if this **InetAddress** (p. 2016) is Multicast and has Organization scope.

## Static Public Member Functions

- static **InetAddress** **getByAddress** (const unsigned char \*bytes, int numBytes)

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 2016) instance.

- static **InetAddress** **getByAddress** (const std::string &hostname, const unsigned char \*bytes, int numBytes)

Given a host name and IPAddress return a new **InetAddress** (p. 2016).

- static **InetAddress** **getLocalHost** ()

Gets an **InetAddress** (p. 2016) that is the local host address.

## Protected Member Functions

- **InetAddress** ()
- **InetAddress** (const unsigned char \*ipAddress, int numBytes)
- **InetAddress** (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

## Static Protected Member Functions

- static unsigned int **bytesToInt** (const unsigned char \*bytes, int start)

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

- static **InetAddress** **getAnyAddress** ()
- static **InetAddress** **getLoopbackAddress** ()

## Protected Attributes

- std::string **hostname**
- bool **reached**
- decaf::lang::ArrayPointer< unsigned char > **addressBytes**

## Static Protected Attributes

- static const unsigned char **loopbackBytes** [4]
- static const unsigned char **anyBytes** [4]

## 6.396.1 Detailed Description

Represents an IP address.

**Since:**

1.0

## 6.396.2 Constructor & Destructor Documentation

**6.396.2.1** `decaf::net::InetAddress::InetAddress ()` [protected]

**6.396.2.2** `decaf::net::InetAddress::InetAddress (const unsigned char * ipAddress, int numBytes)` [protected]

**6.396.2.3** `decaf::net::InetAddress::InetAddress (const std::string & hostname, const unsigned char * ipAddress, int numBytes)` [protected]

**6.396.2.4** `virtual decaf::net::InetAddress::~~InetAddress ()` [virtual]

## 6.396.3 Member Function Documentation

**6.396.3.1** `static unsigned int decaf::net::InetAddress::bytesToInt (const unsigned char * bytes, int start)` [static, protected]

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

**Parameters:**

*bytes* The array of bytes to convert to an int.

*start* The index in the array to treat as the high order byte.

**Returns:**

an unsigned int that represents the address value.

**6.396.3.2** `virtual decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::getAddress () const` [virtual]

Returns the Raw IP address in Network byte order. The returned address is a copy of the bytes contained in this **InetAddress** (p.2016).

**Returns:**

and ArrayPointer containing the raw bytes of the network address.

**6.396.3.3**    `static InetAddress decaf::net::InetAddress::getAnyAddress ()` [static, protected]

**6.396.3.4**    `static InetAddress decaf::net::InetAddress::getByAddress (const std::string & hostname, const unsigned char * bytes, int numBytes)` [static]

Given a host name and IP Address return a new **InetAddress** (p. 2016). There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

**Returns:**

a copy of an **InetAddress** (p. 2016) that represents the given byte array address.

**Exceptions:**

*UnknownHostException* (p. 3907) if the address array length is invalid.

**6.396.3.5**    `static InetAddress decaf::net::InetAddress::getByAddress (const unsigned char * bytes, int numBytes)` [static]

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 2016) instance. An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

**Returns:**

a copy of an **InetAddress** (p. 2016) that represents the given byte array address.

**Exceptions:**

*UnknownHostException* (p. 3907) if the address array length is invalid.

**6.396.3.6**    `virtual std::string decaf::net::InetAddress::getHostAddress ()` const [virtual]

Returns a textual representation of the IP Address.

**Returns:**

the string form of the IP Address.

**6.396.3.7**    `virtual std::string decaf::net::InetAddress::getHostName ()` const [virtual]

Get the host name associated with this **InetAddress** (p. 2016) instance. If a host name was set upon construction then that value is returned, otherwise a reverse name lookup will be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

**Returns:**

the name of the host associated with this set IP Address.

**6.396.3.8 static InetAddress decaf::net::InetAddress::getLocalHost () [static]**

Gets an **InetAddress** (p. 2016) that is the local host address. If the localhost value cannot be resolved than the **InetAddress** (p. 2016) for Loopback is returned.

**Returns:**

a new **InetAddress** (p. 2016) object that contains the local host address.

**Exceptions:**

*UnknownHostException* (p. 3907) if the address for local host is not found.

**6.396.3.9 static InetAddress decaf::net::InetAddress::getLoopbackAddress () [static, protected]****6.396.3.10 virtual bool decaf::net::InetAddress::isAnyLocalAddress () const [inline, virtual]**

Check if this **InetAddress** (p. 2016) is a valid wildcard address.

**Returns:**

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 2012).

**6.396.3.11 virtual bool decaf::net::InetAddress::isLinkLocalAddress () const [inline, virtual]**

Check if this **InetAddress** (p. 2016) is a valid link local address.

**Returns:**

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 2012).

**6.396.3.12 virtual bool decaf::net::InetAddress::isLoopbackAddress () const [inline, virtual]**

Check if this **InetAddress** (p. 2016) is a valid loopback address.

**Returns:**

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 2012).

**6.396.3.13** `virtual bool decaf::net::InetAddress::isMCGlobal () const [inline, virtual]`

Check if this **InetAddress** (p. 2016) is Multicast and has Global scope.

**Returns:**

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2013).

**6.396.3.14** `virtual bool decaf::net::InetAddress::isMCLinkLocal () const [inline, virtual]`

Check if this **InetAddress** (p. 2016) is Multicast and has Link Local scope.

**Returns:**

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2013).

**6.396.3.15** `virtual bool decaf::net::InetAddress::isMCNodeLocal () const [inline, virtual]`

Check if this **InetAddress** (p. 2016) is Multicast and has Node Local scope.

**Returns:**

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2013).

**6.396.3.16** `virtual bool decaf::net::InetAddress::isMCOrgLocal () const [inline, virtual]`

Check if this **InetAddress** (p. 2016) is Multicast and has Organization scope.

**Returns:**

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2013).

**6.396.3.17** `virtual bool decaf::net::InetAddress::isMCSiteLocal () const [inline, virtual]`

Check if this **InetAddress** (p. 2016) is Multicast and has Site Local scope.

**Returns:**

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2013).

**6.396.3.18**    `virtual bool decaf::net::InetAddress::isMulticastAddress () const`  
                  `[inline, virtual]`

Check if this **InetAddress** (p. 2016) is a valid Multicast address.

**Returns:**

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p. 2014).

**6.396.3.19**    `virtual bool decaf::net::InetAddress::isSiteLocalAddress () const`  
                  `[inline, virtual]`

Check if this **InetAddress** (p. 2016) is a valid site local address.

**Returns:**

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 2014).

**6.396.3.20**    `virtual std::string decaf::net::InetAddress::toString () const`    `[virtual]`

Returns a string representation of the **InetAddress** (p. 2016) in the form 'hostname / ipaddress'.  
If the hostname is not resolved than it appears as empty.

**Returns:**

string value of this **InetAddress** (p. 2016).

## 6.396.4    Field Documentation

**6.396.4.1**    `decaf::lang::ArrayPointer<unsigned char> de-`  
                  `caf::net::InetAddress::addressBytes`    `[mutable, protected]`

**6.396.4.2**    `const unsigned char decaf::net::InetAddress::anyBytes[4]`    `[static,`  
                  `protected]`

**6.396.4.3**    `std::string decaf::net::InetAddress::hostname`    `[mutable, protected]`

**6.396.4.4**    `const unsigned char decaf::net::InetAddress::loopbackBytes[4]`    `[static,`  
                  `protected]`

**6.396.4.5**    `bool decaf::net::InetAddress::reached`    `[mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetAddress.h`

## 6.397 decaf::net::InetSocketAddress Class Reference

#include <src/main/decaf/net/InetSocketAddress.h> Inheritance diagram for decaf::net::InetSocketAddress:

### Public Member Functions

- **InetSocketAddress** ()
- virtual **~InetSocketAddress** ()

### 6.397.1 Constructor & Destructor Documentation

**6.397.1.1** decaf::net::InetSocketAddress::InetSocketAddress ()

**6.397.1.2** virtual decaf::net::InetSocketAddress::~~InetSocketAddress () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**InetSocketAddress.h**

## 6.398 inflate\_\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/inflate.h>
```

### Data Fields

- **inflate\_\_mode** mode
- int **last**
- int **wrap**
- int **havedict**
- int **flags**
- unsigned **dmax**
- unsigned long **check**
- unsigned long **total**
- **gz\_headerp** head
- unsigned **wbits**
- unsigned **wsize**
- unsigned **whave**
- unsigned **wnext**
- unsigned char FAR \* **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code** const FAR \* **lencode**
- **code** const FAR \* **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**
- **code** FAR \* **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- **code** codes [ENOUGH]
- int **sane**
- int **back**
- unsigned **was**



## 6.398.1 Field Documentation

- 6.398.1.1 `int inflate__state::back`
- 6.398.1.2 `unsigned inflate__state::bits`
- 6.398.1.3 `unsigned long inflate__state::check`
- 6.398.1.4 `code inflate__state::codes[ENOUGH]`
- 6.398.1.5 `unsigned inflate__state::distbits`
- 6.398.1.6 `code const FAR* inflate__state::distcode`
- 6.398.1.7 `unsigned inflate__state::dmax`
- 6.398.1.8 `unsigned inflate__state::extra`
- 6.398.1.9 `int inflate__state::flags`
- 6.398.1.10 `unsigned inflate__state::have`
- 6.398.1.11 `int inflate__state::havedict`
- 6.398.1.12 `gz_headerp inflate__state::head`
- 6.398.1.13 `unsigned long inflate__state::hold`
- 6.398.1.14 `int inflate__state::last`
- 6.398.1.15 `unsigned inflate__state::lenbits`
- 6.398.1.16 `code const FAR* inflate__state::lencode`
- 6.398.1.17 `unsigned inflate__state::length`
- 6.398.1.18 `unsigned short inflate__state::lens[320]`
- 6.398.1.19 `inflate__mode inflate__state::mode`
- 6.398.1.20 `unsigned inflate__state::ncode`
- 6.398.1.21 `unsigned inflate__state::ndist`
- 6.398.1.22 `code FAR* inflate__state::next`
- 6.398.1.23 `unsigned inflate__state::nlen`
- 6.398.1.24 `unsigned inflate__state::offset`
- 6.398.1.25 `int inflate__state::sane`
- 6.398.1.26 `unsigned long inflate__state::total`
- 6.398.1.27 `unsigned inflate__state::was`
- 6.398.1.28 `unsigned inflate__state::wbits`
- 6.398.1.29 `unsigned inflate__state::whave`
- 6.398.1.30 `unsigned char FAR* inflate__state::window`

- `src/main/decaf/internal/util/zip/inflate.h`

## 6.399 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see specification).

```
#include <src/main/decaf/util/zip/Inflater.h>
```

### Public Member Functions

- **Inflater** ()  
*Creates a new decompressor.*
- **Inflater** (bool nowrap)  
*Creates a new decompressor.*
- virtual **~Inflater** ()
- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for decompression.*
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for decompression.*
- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets input data for decompression.*
- int **getRemaining** () const  
*Returns the total number of bytes remaining in the input buffer.*
- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the preset dictionary to the given array of bytes.*
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the preset dictionary to the given array of bytes.*
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the preset dictionary to the given array of bytes.*

- bool **needsInput** () const
- bool **needsDictionary** () const
- void **finish** ()

*When called, indicates that decompression should end with the current contents of the input buffer.*

- bool **finished** () const
- int **inflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

*Uncompresses bytes into specified buffer.*

- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

*Uncompresses bytes into specified buffer.*

- int **inflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )

*Uncompresses bytes into specified buffer.*

- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )

*Resets deflater so that a new set of input data can be processed.*

- void **end** ()

*Closes the decompressor and discards any unprocessed input.*

### 6.399.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see [specification](#)). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 2035) and its descendants.

The typical usage of a **Inflater** (p. 2027) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 2035).

See also:

**InflaterInputStream** (p. 2035)  
**Deflater** (p. 1706)

Since:

1.0

## 6.399.2 Constructor & Destructor Documentation

### 6.399.2.1 decaf::util::zip::Inflater::Inflater ()

Creates a new decompressor. This constructor defaults the inflater to use the ZLIB header and checksum fields.

### 6.399.2.2 decaf::util::zip::Inflater::Inflater (bool nowrap)

Creates a new decompressor. If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

### 6.399.2.3 virtual decaf::util::zip::Inflater::~Inflater () [virtual]

## 6.399.3 Member Function Documentation

### 6.399.3.1 void decaf::util::zip::Inflater::end ()

Closes the decompressor and discards any unprocessed input. This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p. 2027) object is undefined.

### 6.399.3.2 void decaf::util::zip::Inflater::finish ()

When called, indicates that decompression should end with the current contents of the input buffer.

### 6.399.3.3 bool decaf::util::zip::Inflater::finished () const

#### Returns:

true if the end of the compressed data output stream has been reached.

### 6.399.3.4 long long decaf::util::zip::Inflater::getAdler () const throw (decaf::lang::exceptions::IllegalStateException )

#### Returns:

the ADLER-32 value of the uncompressed data.

#### Exceptions:

*IllegalStateException* if in the end state.

**6.399.3.5** `long long decaf::util::zip::Inflater::getBytesRead () const throw ( decaf::lang::exceptions::IllegalStateException )`

**Returns:**

the total number of compressed bytes input so far.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.399.3.6** `long long decaf::util::zip::Inflater::getBytesWritten () const throw ( decaf::lang::exceptions::IllegalStateException )`

**Returns:**

the total number of decompressed bytes output so far.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.399.3.7** `int decaf::util::zip::Inflater::getRemaining () const`

Returns the total number of bytes remaining in the input buffer. This can be used to find out what bytes still remain in the input buffer after decompression has finished.

**Returns:**

the total number of bytes remaining in the input buffer

**6.399.3.8** `int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )`

Uncompresses bytes into specified buffer. Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2032) or **needsDictionary()** (p. 2032) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2029) can be used to get the Adler-32 value of the dictionary required.

**Parameters:**

*buffer* The Buffer to write the compressed data to.

**Exceptions:**

*IllegalStateException* if in the end state.

*DataFormatException* (p. 1555) if the compressed data format is invalid.

**6.399.3.9** int decaf::util::zip::Inflater::inflate (std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

Uncompresses bytes into specified buffer. Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2032) or **needsDictionary()** (p. 2032) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2029) can be used to get the Adler-32 value of the dictionary required.

**Parameters:**

*buffer* The Buffer to write the compressed data to.

*offset* The position in the Buffer to start writing at.

*length* The maximum number of byte of data to write.

**Exceptions:**

*IllegalStateException* if in the end state.

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*DataFormatException* (p. 1555) if the compressed data format is invalid.

**6.399.3.10** int decaf::util::zip::Inflater::inflate (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

Uncompresses bytes into specified buffer. Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2032) or **needsDictionary()** (p. 2032) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2029) can be used to get the Adler-32 value of the dictionary required.

**Parameters:**

*buffer* The Buffer to write the compressed data to.

*size* The size of the buffer passed in.

*offset* The position in the Buffer to start writing at.

*length* The maximum number of byte of data to write.

**Exceptions:**

*NullPointerException* if buffer is NULL.

*IllegalStateException* if in the end state.

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*DataFormatException* (p. 1555) if the compressed data format is invalid.

**6.399.3.11** `bool decaf::util::zip::Inflater::needsDictionary () const`**Returns:**

true if a preset dictionary is needed for decompression.

**6.399.3.12** `bool decaf::util::zip::Inflater::needsInput () const`**Returns:**

true if the input data buffer is empty and `setInput()` (p. 2034) should be called in order to provide more input

**6.399.3.13** `void decaf::util::zip::Inflater::reset () throw ( decaf::lang::exceptions::IllegalStateException )`

Resets deflater so that a new set of input data can be processed. Keeps current decompression level and strategy settings.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.399.3.14** `void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )`

Sets the preset dictionary to the given array of bytes. Should be called when `inflate()` (p. 2031) returns 0 and `needsDictionary()` (p. 2032) returns true indicating that a preset dictionary is required. The method `getAdler()` (p. 2029) can be used to get the Adler-32 value of the dictionary needed.

**Parameters:**

*buffer* The Buffer to read in for decompression.

**Exceptions:**

*IllegalStateException* if in the end state.

*IllegalArgumentException* if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.15** `void decaf::util::zip::Inflater::setDictionary (const std::vector< unsigned char > & buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the preset dictionary to the given array of bytes. Should be called when `inflate()` (p. 2031) returns 0 and `needsDictionary()` (p. 2032) returns true indicating that a preset dictionary is required. The method `getAdler()` (p. 2029) can be used to get the Adler-32 value of the dictionary needed.



**Parameters:**

*buffer* The Buffer to read in for decompression.  
*offset* The position in the Buffer to start reading from.  
*length* The number of bytes to read from the input buffer.

**Exceptions:**

*IndexOutOfBoundsException* if the offset + length > size of the buffer.  
*IllegalStateException* if in the end state.  
*IllegalArgumentException* if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.16** void decaf::util::zip::Inflater::setDictionary (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Sets the preset dictionary to the given array of bytes. Should be called when **inflate()** (p. 2031) returns 0 and **needsDictionary()** (p. 2032) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 2029) can be used to get the Adler-32 value of the dictionary needed.

**Parameters:**

*buffer* The Buffer to read in for decompression.  
*size* The size of the buffer passed in.  
*offset* The position in the Buffer to start reading from.  
*length* The number of bytes to read from the input buffer.

**Exceptions:**

*NullPointerException* if buffer is NULL.  
*IndexOutOfBoundsException* if the offset + length > size of the buffer.  
*IllegalStateException* if in the end state.  
*IllegalArgumentException* if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.17** void decaf::util::zip::Inflater::setInput (const std::vector< unsigned char > & *buffer*) throw ( decaf::lang::exceptions::IllegalStateException )

Sets input data for decompression. This should be called whenever **needsInput()** (p. 2032) returns true indicating that more input data is required.

**Parameters:**

*buffer* The Buffer to read in for decompression.

**Exceptions:**

*IllegalStateException* if in the end state.

**6.399.3.18** void decaf::util::zip::Inflater::setInput (const std::vector< unsigned char > & *buffer*, int *offset*, int *length*) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for decompression. This should be called whenever **needsInput()** (p. 2032) returns true indicating that more input data is required.

**Parameters:**

*buffer* The Buffer to read in for decompression.

*offset* The position in the Buffer to start reading from.

*length* The number of bytes to read from the input buffer.

**Exceptions:**

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*IllegalStateException* if in the end state.

**6.399.3.19** void decaf::util::zip::Inflater::setInput (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for decompression. This should be called whenever **needsInput()** (p. 2032) returns true indicating that more input data is required.

**Parameters:**

*buffer* The Buffer to read in for decompression.

*size* The size of the buffer passed in.

*offset* The position in the Buffer to start reading from.

*length* The number of bytes to read from the input buffer.

**Exceptions:**

*NullPointerException* if buffer is NULL.

*IndexOutOfBoundsException* if the offset + length > size of the buffer.

*IllegalStateException* if in the end state.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

## 6.400 decaf::util::zip::InflaterInputStream Class Reference

A `FilterInputStream` that decompresses data read from the wrapped `InputStream` instance.

#include <src/main/decaf/util/zip/InflaterInputStream.h> Inheritance diagram for decaf::util::zip::InflaterInputStream:

### Public Member Functions

- **InflaterInputStream** (decaf::io::InputStream \*inputStream, bool own=false)  
*Create an instance of this class with a default inflater and buffer size.*
- **InflaterInputStream** (decaf::io::InputStream \*inputStream, Inflater \*inflater, bool own=false)  
*Creates a new **InflaterInputStream** (p. 2035) with a user supplied **Inflater** (p. 2027) and a default buffer size.*
- **InflaterInputStream** (decaf::io::InputStream \*inputStream, Inflater \*inflater, int bufferSize, bool own=false)  
*Creates a new **DeflateOutputStream** with a user supplied **Inflater** (p. 2027) and specified buffer size.*
- virtual ~**InflaterInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.*  
**Returns:**  
*the number of bytes available on this input stream.*  
**Exceptions:**  
***IOException** (p. 2142) if an I/O error occurs.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2043) freeing any resources that might have been acquired during the lifetime of this stream.  
The default implementation of this method does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.  
The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters:**

*num* The number of bytes to skip.

**Returns:**

*total bytes skipped*

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

**Parameters:**

*readLimit* The max bytes read before marked position is invalid.

- virtual void **reset** () throw ( decaf::io::IOException )

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2142).

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns:**

*true if this stream instance supports marks*

## Protected Member Functions

- virtual void **fill** () throw ( decaf::io::IOException )

Fills the input buffer with the next chunk of data.

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int **length**) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

## Protected Attributes

- **Inflater \* inflater**  
*The **Inflater** (p. 2027) instance to use.*
- std::vector< unsigned char > **buff**  
*The buffer to hold chunks of data read from the stream before inflation.*
- int **length**  
*The amount of data currently stored in the input buffer.*
- bool **ownInflater**
- bool **atEOF**

## Static Protected Attributes

- static const int **DEFAULT\_BUFFER\_SIZE**

### 6.400.1 Detailed Description

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

**Since:**

1.0

### 6.400.2 Constructor & Destructor Documentation

#### 6.400.2.1 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream \* *inputStream*, bool *own* = false)

Create an instance of this class with a default inflater and buffer size.

**Parameters:**

- inputStream* The InputStream instance to wrap.
- own* Should this Filter take ownership of the InputStream pointer (defaults to false).

#### 6.400.2.2 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream \* *inputStream*, Inflater \* *inflater*, bool *own* = false)

Creates a new **InflaterInputStream** (p. 2035) with a user supplied **Inflater** (p. 2027) and a default buffer size. When the user supplied a **Inflater** (p. 2027) instance the **InflaterInputStream** (p. 2035) does not take ownership of the **Inflater** (p. 2027) pointer, the caller is still responsible for deleting the **Inflater** (p. 2027).

**Parameters:**

- inputStream* The InputStream instance to wrap.
- inflater* The user supplied **Inflater** (p. 2027) to use for decompression. (
- own* Should this filter take ownership of the InputStream pointer (default is false).

**Exceptions:**

- NullPointerException* if the **Inflater** (p. 2027) given is NULL.

#### 6.400.2.3 decaf::util::zip::InflaterInputStream::InflaterInputStream (decaf::io::InputStream \* inputStream, Inflater \* inflater, int bufferSize, bool own = false)

Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 2027) and specified buffer size. When the user supplied a **Inflater** (p. 2027) instance the **InflaterInputStream** (p. 2035) does not take ownership of the **Inflater** (p. 2027) pointer, the caller is still responsible for deleting the **Inflater** (p. 2027).

**Parameters:**

- inputStream* The InputStream instance to wrap.
- inflater* The user supplied **Inflater** (p. 2027) to use for decompression.
- bufferSize* The size of the input buffer.
- own* Should this filter take ownership of the InputStream pointer (default is false).

**Exceptions:**

- NullPointerException* if the **Inflater** (p. 2027) given is NULL.
- IllegalArgumentException* if the bufferSize value is zero.

#### 6.400.2.4 virtual decaf::util::zip::InflaterInputStream::~~InflaterInputStream () [virtual]

### 6.400.3 Member Function Documentation

#### 6.400.3.1 virtual int decaf::util::zip::InflaterInputStream::available () const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns:**

- the number of bytes available on this input stream.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from **decaf::io::FilterInputStream** (p. 1896).

**6.400.3.2 virtual void decaf::util::zip::InflaterInputStream::close () throw ( decaf::io::IOException ) [virtual]**

Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing. Closes any resources associated with this **InflaterInputStream** (p. 2035).

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

**6.400.3.3 virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]**

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

**6.400.3.4 virtual int decaf::util::zip::InflaterInputStream::doReadByte () throw ( decaf::io::IOException ) [protected, virtual]**

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

**6.400.3.5 virtual void decaf::util::zip::InflaterInputStream::fill () throw ( decaf::io::IOException ) [protected, virtual]**

Fills the input buffer with the next chunk of data.

**Exceptions:**

*IOException* if an I/O error occurs.

**6.400.3.6 virtual void decaf::util::zip::InflaterInputStream::mark (int *readLimit*) [virtual]**

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

**Parameters:**

*readLimit* The max bytes read before marked position is invalid.

Does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1897).

**6.400.3.7 virtual bool decaf::util::zip::InflaterInputStream::markSupported () const [virtual]**

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns:**

true if this stream instance supports marks

Always returns false.

Reimplemented from **decaf::io::FilterInputStream** (p. 1898).

**6.400.3.8 virtual void decaf::util::zip::InflaterInputStream::reset () throw ( decaf::io::IOException ) [virtual]**

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2142).

**Exceptions:**

**IOException** (p. 2142) if an I/O error occurs.

Always throws an **IOException** when called.

Reimplemented from **decaf::io::FilterInputStream** (p. 1898).



**6.400.3.9** `virtual long long decaf::util::zip::InflaterInputStream::skip`  
`(long long num) throw ( decaf::io::IOException,`  
`decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

*num* The number of bytes to skip.

#### Returns:

total bytes skipped

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

*UnsupportedOperationException* if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from **decaf::io::FilterInputStream** (p. 1899).

### 6.400.4 Field Documentation

**6.400.4.1** `bool decaf::util::zip::InflaterInputStream::atEOF` [protected]

**6.400.4.2** `std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff`  
[protected]

The buffer to hold chunks of data read from the stream before inflation.

**6.400.4.3** `const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE` [static, protected]

**6.400.4.4** `Inflater* decaf::util::zip::InflaterInputStream::inflater` [protected]

The **Inflater** (p. 2027) instance to use.

**6.400.4.5** `int decaf::util::zip::InflaterInputStream::length` [protected]

The amount of data currently stored in the input buffer.

**6.400.4.6**   `bool decaf::util::zip::InflaterInputStream::ownInflater`   [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/InflaterInputStream.h`

## 6.401 decaf::io::InputStream Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

#include <src/main/decaf/io/InputStream.h> Inheritance diagram for decaf::io::InputStream:

### Public Member Functions

- **InputStream** ()
- virtual **~InputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.*
- virtual void **mark** (int readLimit)  
*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Repositions this stream to the position at the time the mark method was last called on this input stream.*
- virtual bool **markSupported** () const  
*Determines if this input stream supports the mark and reset methods.*
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.*
- virtual int **read** () throw ( decaf::io::IOException )  
*Reads a single byte from the buffer.*
- virtual int **read** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reads up to size bytes of data from the input stream into an array of bytes.*
- virtual int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Reads up to length bytes of data from the input stream into an array of bytes.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

## Protected Member Functions

- virtual int **doReadByte** ()=0 throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.401.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

**Since:**

1.0

## 6.401.2 Constructor & Destructor Documentation

6.401.2.1 `decaf::io::InputStream::InputStream ()`

6.401.2.2 `virtual decaf::io::InputStream::~~InputStream ()` [virtual]

## 6.401.3 Member Function Documentation

6.401.3.1 `virtual int decaf::io::InputStream::available () const throw ( decaf::io::IOException )` [inline, virtual]

Indicates the number of bytes available. The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

### Returns:

the number of bytes available on this input stream.

### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented in `decaf::internal::io::StandardInputStream` (p. 3582), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2883), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3748), `decaf::io::BlockingByteArrayInputStream` (p. 840), `decaf::io::BufferedInputStream` (p. 936), `decaf::io::ByteArrayInputStream` (p. 1023), `decaf::io::FilterInputStream` (p. 1896), `decaf::io::PushbackInputStream` (p. 3143), and `decaf::util::zip::InflaterInputStream` (p. 2038).

6.401.3.2 `virtual void decaf::io::InputStream::close () throw ( decaf::io::IOException )` [virtual]

Closes the **InputStream** (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream. The default implementation of this method does nothing.

Implements `decaf::io::Closeable` (p. 1149).

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2883), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3748), `decaf::io::BlockingByteArrayInputStream` (p. 841), `decaf::io::BufferedInputStream` (p. 937), `decaf::io::FilterInputStream` (p. 1897), and `decaf::util::zip::InflaterInputStream` (p. 2039).

6.401.3.3 `virtual int decaf::io::InputStream::doReadArray (unsigned char * buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented in `decaf::io::FilterInputStream` (p. 1897).

**6.401.3.4** `virtual int decaf::io::InputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented in `activemq::io::LoggingInputStream` (p. 2399), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2883), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3749), `decaf::io::BlockingByteArrayInputStream` (p. 841), `decaf::io::BufferedInputStream` (p. 937), `decaf::io::ByteArrayInputStream` (p. 1024), `decaf::io::FilterInputStream` (p. 1897), `decaf::io::PushbackInputStream` (p. 3144), `decaf::util::zip::CheckedInputStream` (p. 1138), and `decaf::util::zip::InflaterInputStream` (p. 2039).

**6.401.3.5** `virtual int decaf::io::InputStream::doReadByte () throw ( decaf::io::IOException )` [protected, pure virtual]

Implemented in `activemq::io::LoggingInputStream` (p. 2400), `decaf::internal::io::StandardInputStream` (p. 3583), `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` (p. 2884), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3749), `decaf::io::BlockingByteArrayInputStream` (p. 841), `decaf::io::BufferedInputStream` (p. 937), `decaf::io::ByteArrayInputStream` (p. 1024), `decaf::io::FilterInputStream` (p. 1897), `decaf::io::PushbackInputStream` (p. 3144), `decaf::util::zip::CheckedInputStream` (p. 1138), and `decaf::util::zip::InflaterInputStream` (p. 2039).

**6.401.3.6** `virtual void decaf::io::InputStream::lock () throw ( decaf::lang::exceptions::RuntimeException )` [inline, virtual]

Locks the object.

#### Exceptions:

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3701).

**6.401.3.7** `virtual void decaf::io::InputStream::mark (int readLimit)` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters:

*readLimit* The max bytes read before marked position is invalid.

Reimplemented in `decaf::io::BufferedInputStream` (p. 937), `decaf::io::ByteArrayInputStream` (p. 1024), `decaf::io::FilterInputStream` (p. 1897),

**decaf::io::PushbackInputStream** (p. 3144), and **decaf::util::zip::InflaterInputStream** (p. 2039).

**6.401.3.8** `virtual bool decaf::io::InputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns:

true if this stream instance supports marks

Reimplemented in **decaf::io::BufferedInputStream** (p. 937), **decaf::io::ByteArrayInputStream** (p. 1024), **decaf::io::FilterInputStream** (p. 1898), **decaf::io::PushbackInputStream** (p. 3144), and **decaf::util::zip::InflaterInputStream** (p. 2040).

**6.401.3.9** `virtual void decaf::io::InputStream::notify ()  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

***RuntimeException*** if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3702).

**6.401.3.10** `virtual void decaf::io::InputStream::notifyAll ()  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

***RuntimeException*** if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.401.3.11** `virtual int decaf::io::InputStream::read (unsigned char * buffer,  
int size, int offset, int length) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException ) [virtual]`

Reads up to *length* bytes of data from the input stream into an array of bytes. An attempt is made to read as many as *length* bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If *length* is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into *b*.

The first byte read is stored into element *b*[*off*], the next one into *b*[*off*+1], and so on. The number of bytes read is, at most, equal to *length*. Let *k* be the number of bytes actually read; these bytes will be stored in elements *b*[*off*] through *b*[*off*+*k*-1], leaving elements *b*[*offset*+*k*] through *b*[*offset*+*length*-1] unaffected.

In every case, elements *b*[0] through *b*[*offset*] and elements *b*[*offset*+*length*] through *b*[*b.length*-1] are unaffected.

This method called the protected virtual method `doReadArrayBounded` which simply calls the method `doReadByte()` (p.2046) repeatedly. If the first such call results in an **IOException** (p.2142), that exception is returned. If any subsequent call to `doReadByte()` (p.2046) results in a **IOException** (p.2142), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

*buffer* The target buffer to write the read in data to.

*size* The size in bytes of the target buffer.

*offset* The position in the buffer to start inserting the read in data.

*length* The maximum number of bytes that should be read into buffer.

#### Returns:

The number of bytes read or -1 if EOF is detected

#### Exceptions:

**IOException** (p.2142) if an I/O error occurs.

**NullPointerException** if buffer passed is NULL.

**IndexOutOfBoundsException** if *length* > *size* - *offset*.

**6.401.3.12** `virtual int decaf::io::InputStream::read (unsigned char  
* buffer, int size) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException ) [virtual]`

Reads up to *size* bytes of data from the input stream into an array of bytes. An attempt is made to read as many as *size* bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.



This method blocks until input data is available, end of file is detected, or an exception is thrown.

If size is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

This method called the protected virtual method `doReadArray` which by default is the same as calling `read( buffer, size, 0, size )`. Subclasses can customize the behavior of this method by overriding the `doReadArray` method to provide a better performing read operation.

**Parameters:**

*buffer* The target buffer to write the read in data to.

*size* The size in bytes of the target buffer.

**Returns:**

The number of bytes read or -1 if EOF is detected

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

*NullPointerException* if buffer passed is NULL.

#### 6.401.3.13 virtual int decaf::io::InputStream::read () throw ( decaf::io::IOException ) [virtual]

Reads a single byte from the buffer. The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

The default implementation of this method calls the **internal** (p. 144) virtual method `doReadByte` which is a pure virtual method and must be overridden by all subclasses.

**Returns:**

The next byte or -1 if the end of stream is reached.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

#### 6.401.3.14 virtual void decaf::io::InputStream::reset () throw ( decaf::io::IOException ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method `markSupported` returns true, then: \* If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method `markSupported` returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2142).

#### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

Reimplemented in **decaf::io::BufferedInputStream** (p. 938), **decaf::io::ByteArrayInputStream** (p. 1025), **decaf::io::FilterInputStream** (p. 1898), **decaf::io::PushbackInputStream** (p. 3145), and **decaf::util::zip::InflaterInputStream** (p. 2040).

**6.401.3.15** `virtual long long decaf::io::InputStream::skip (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

*num* The number of bytes to skip.

#### Returns:

total bytes skipped

#### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2884), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3749), **decaf::io::BlockingByteArrayInputStream** (p. 841), **decaf::io::BufferedInputStream** (p. 938), **decaf::io::ByteArrayInputStream** (p. 1026), **decaf::io::FilterInputStream** (p. 1899), **decaf::io::PushbackInputStream** (p. 3145), **decaf::util::zip::CheckedInputStream** (p. 1139), and **decaf::util::zip::InflaterInputStream** (p. 2041).

**6.401.3.16** `virtual std::string decaf::io::InputStream::toString () const` [virtual]

Output a String representation of this object. The default version of this method just prints the Class Name.

**Returns:**

a string representation of the object.

**6.401.3.17** `virtual bool decaf::io::InputStream::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

**6.401.3.18** `virtual void decaf::io::InputStream::unlock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).

**6.401.3.19** `virtual void decaf::io::InputStream::wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

**Exceptions:**

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.401.3.20** `virtual void decaf::io::InputStream::wait (long long  
    millisecs) throw ( decaf::lang::exceptions::RuntimeException,  
    decaf::lang::exceptions::IllegalMonitorStateException,  
    decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

**6.401.3.21** `virtual void decaf::io::InputStream::wait () throw  
    ( decaf::lang::exceptions::RuntimeException,  
    decaf::lang::exceptions::IllegalMonitorStateException,  
    decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStream.h`

## 6.402 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p. 2053) is a bridge from byte streams to character streams.

#include <src/main/decaf/io/InputStreamReader.h> Inheritance diagram for decaf::io::InputStreamReader:

### Public Member Functions

- **InputStreamReader** (**InputStream** \*stream, bool own=false)  
*Create a new **InputStreamReader** (p. 2053) that wraps the given **InputStream** (p. 2043).*
- virtual ~**InputStreamReader** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*

### Protected Member Functions

- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ).*
- virtual void **checkClosed** () const throw ( decaf::io::IOException )

#### 6.402.1 Detailed Description

An **InputStreamReader** (p. 2053) is a bridge from byte streams to character streams. For top efficiency, consider wrapping an **InputStreamReader** (p. 2053) within a **BufferedReader**. For example:

```
BufferedReader* in = new BufferedReader( new InputStreamReader (p. 2053)( System.in, false ), true );
```

See also:

**OutputStreamWriter** (p. 2915)

Since:

1.0

## 6.402.2 Constructor & Destructor Documentation

### 6.402.2.1 `decaf::io::InputStreamReader::InputStreamReader (InputStream * stream, bool own = false)`

Create a new **InputStreamReader** (p. 2053) that wraps the given **InputStream** (p. 2043).

#### Parameters:

*stream* The **InputStream** (p. 2043) to read from. (cannot be null).

*own* Does this object own the passed **InputStream** (p. 2043) (defaults to false).

#### Exceptions:

*NullPointerException* if the passed **InputStream** (p. 2043) is NULL.

### 6.402.2.2 `virtual decaf::io::InputStreamReader::~~InputStreamReader ()` [virtual]

## 6.402.3 Member Function Documentation

### 6.402.3.1 `virtual void decaf::io::InputStreamReader::checkClosed () const throw ( decaf::io::IOException )` [protected, virtual]

### 6.402.3.2 `virtual void decaf::io::InputStreamReader::close () throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

#### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 1149).

### 6.402.3.3 `virtual int decaf::io::InputStreamReader::doReadArrayBounded (char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Override this method to customize the functionality of the method `read( unsigned char* buffer, int size, int offset, int length )`. All subclasses must override this method to provide the basic **Reader** (p. 3163) functionality.

Implements **decaf::io::Reader** (p. 3164).

### 6.402.3.4 `virtual bool decaf::io::InputStreamReader::ready () const throw ( decaf::io::IOException )` [virtual]

Tells whether this stream is ready to be read.

**Returns:**

True if the next **read()** (p. 3166) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::Reader** (p. 3167).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStreamReader.h`

## 6.403 decaf::internal::nio::IntArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/IntArrayBuffer.h> Inheritance diagram for decaf::internal::nio::IntArrayBuffer:

### Public Member Functions

- **IntArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **IntArrayBuffer** (p. 2056) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **IntArrayBuffer** (int \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **IntArrayBuffer** (p. 2056) object that wraps the given array.*

- **IntArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **IntArrayBuffer** (const IntArrayBuffer &other)

*Create a **IntArrayBuffer** (p. 2056) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**IntArrayBuffer** ()
- virtual int \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the int array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928).*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset into the backing array where index zero starts.*



**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual **IntArrayBuffer** \* **asReadOnlyBuffer** () const

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only int buffer which the caller then owns.

- virtual **IntArrayBuffer** & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **IntArrayBuffer** (p. 2066)

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual **IntArrayBuffer** \* **duplicate** ()

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new int **Buffer** (p. 928) which the caller owns.

- virtual int **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the int at the current position.

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return.

- virtual int **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the int is to be read.

**Returns:**

the int that is located at the given index.

**Exceptions:**

**IndexOutOfBoundsException** if *index* is not smaller than the buffer's limit, or *index* is negative.

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible int array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.*

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns:**

true if, and only if, this buffer is read-only.

- virtual IntBuffer & **put** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given integer into this buffer at the current position, and then increments the position.*

**Parameters:**

*value* The integer value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual IntBuffer & **put** (int index, int value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given ints into this buffer at the given index.*

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.  
*value* The ints to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** - If *index* greater than the buffer's limit minus the size of the type being written, or the *index* is negative.

**ReadOnlyBufferException** (p. 3169) - If this buffer is read-only.

- virtual IntBuffer \* **slice** () const

Creates a new **IntBuffer** (p. 2066) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **IntBuffer** (p. 2066) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **IntArrayBuffer** (p. 2056) as Read-Only.

### 6.403.1 Constructor & Destructor Documentation

**6.403.1.1 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer**  
(int *size*, bool *readOnly* = false) throw (  
decaf::lang::exceptions::IllegalArgumentException )

Creates a **IntArrayBuffer** (p. 2056) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

*size* The size of the array, this is the limit we read and write to.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

**IllegalArgumentException** if the capacity value is negative.

**6.403.1.2 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer** (int \*  
*array*, int *size*, int *offset*, int *length*, bool *readOnly* =  
false) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **IntArrayBuffer** (p. 2056) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The actual array to wrap.

*size* The size of the given array.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

**6.403.1.3** `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer`  
 (const decaf::lang::Pointer< ByteArrayAdapter > &  
*array*, int *offset*, int *length*, bool *readOnly* = false)  
 throw ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset. The capacity and limit of the new **IntArrayBuffer** (p.2056) will be that of the remaining capacity of the passed buffer.

#### Parameters:

*array* The ByteArrayAdapter to wrap.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if array is NULL

*IndexOutOfBoundsException* if offset + length is greater than array size.

**6.403.1.4** `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer` (const  
 IntArrayBuffer & *other*)

Create a **IntArrayBuffer** (p.2056) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters:

*other* The **IntArrayBuffer** (p.2056) this one is to mirror.

**6.403.1.5** `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer` ()  
 [virtual]

### 6.403.2 Member Function Documentation

**6.403.2.1** `virtual int* decaf::internal::nio::IntArrayBuffer::array` () throw  
 ( decaf::lang::exceptions::UnsupportedOperationException,  
 decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2069).

**6.403.2.2** `virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2069).

**6.403.2.3** `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer  
( ) const [virtual]`

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 2069).

#### 6.403.2.4 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact () throw ( decaf::nio::ReadOnlyBufferException ) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 932) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 932) - 1 is copied to index `n = limit() - 1 - p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

##### Returns:

a reference to this **IntBuffer** (p. 2066)

##### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2070).

#### 6.403.2.5 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate () [virtual]`

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

##### Returns:

a new int **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 2070).

#### 6.403.2.6 `virtual int decaf::internal::nio::IntArrayBuffer::get (int index) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Absolute get method.

Reads the value at the given index.

##### Parameters:

*index* The index in the **Buffer** (p. 928) where the int is to be read.

##### Returns:

the int that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::IntBuffer** (p. 2071).

**6.403.2.7** `virtual int decaf::internal::nio::IntArrayBuffer::get () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the int at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implements **decaf::nio::IntBuffer** (p. 2072).

**6.403.2.8** `virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::IntBuffer** (p. 2072).

**6.403.2.9** `virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 931).

**6.403.2.10** `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int index, int value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given ints into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The ints to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* - If index greater than the buffer's limit minus the size of the type being written, or the index is negative.

*ReadOnlyBufferException* (p. 3169) - If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2072).

**6.403.2.11** `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given integer into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The integer value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2073).

**6.403.2.12** `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **IntArrayBuffer** (p. 2056) as Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.403.2.13** `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const [virtual]`

Creates a new **IntBuffer** (p. 2066) whose content is a shared subsequence of this buffer's content.



The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **IntBuffer** (p.2066) which the caller owns.

Implements **decaf::nio::IntBuffer** (p.2075).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**IntArrayBuffer.h**

## 6.404 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:.

#include <src/main/decaf/nio/IntBuffer.h>Inheritance diagram for decaf::nio::IntBuffer:

### Public Member Functions

- virtual **~IntBuffer** ()
- virtual std::string **toString** () const
- virtual int \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the int array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **IntBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only int buffer that shares this buffer's content.*
- virtual **IntBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **IntBuffer** \* **duplicate** ()=0  
*Creates a new int buffer that shares this buffer's content.*
- virtual int **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual int **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **IntBuffer** & **get** (std::vector< int > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **IntBuffer** & **get** (int \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible int array.*
- **IntBuffer** & **put** (**IntBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the ints remaining in the given source buffer into this buffer.*

- **IntBuffer** & **put** (const int \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers ints into this buffer from the given source array.*

- **IntBuffer** & **put** (std::vector< int > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source ints array into this buffer.*

- virtual **IntBuffer** & **put** (int value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given integer into this buffer at the current position, and then increments the position.*

- virtual **IntBuffer** & **put** (int index, int value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given ints into this buffer at the given index.*

- virtual **IntBuffer** \* **slice** () const =0

*Creates a new **IntBuffer** (p. 2066) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **IntBuffer** &value) const

- virtual bool **equals** (const **IntBuffer** &value) const

- virtual bool **operator==** (const **IntBuffer** &value) const

- virtual bool **operator<** (const **IntBuffer** &value) const

## Static Public Member Functions

- static **IntBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new Double buffer.*

- static **IntBuffer** \* **wrap** (int \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **IntBuffer** (p. 2066).*

- static **IntBuffer** \* **wrap** (std::vector< int > &buffer)

*Wraps the passed STL int Vector in a **IntBuffer** (p. 2066).*

## Protected Member Functions

- **IntBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **IntBuffer** (p. 2066) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.404.1 Detailed Description

This class defines four categories of operations upon int buffers: o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.404.2 Constructor & Destructor Documentation

- 6.404.2.1** decaf::nio::IntBuffer::IntBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **IntBuffer** (p. 2066) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

#### Parameters:

*capacity* The size and limit of the **Buffer** (p. 928) in integers.

#### Exceptions:

*IllegalArgumentException* if capacity is negative.

- 6.404.2.2** virtual decaf::nio::IntBuffer::~~IntBuffer () [inline, virtual]

### 6.404.3 Member Function Documentation

- 6.404.3.1** static IntBuffer\* decaf::nio::IntBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters:

*capacity* The size of the Double buffer in integers.

#### Returns:

the **IntBuffer** (p. 2066) that was allocated, caller owns.

### 6.404.3.2 virtual int\* decaf::nio::IntBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the int array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns:

the array that backs this **Buffer** (p. 928).

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2060).

### 6.404.3.3 virtual int decaf::nio::IntBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns:

The offset into the backing array where index zero starts.

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2061).

### 6.404.3.4 virtual IntBuffer\* decaf::nio::IntBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns:

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2061).

**6.404.3.5** **virtual IntBuffer& decaf::nio::IntBuffer::compact () throw (**  
**ReadOnlyBufferException )** [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index **limit()** (p. 932) - 1 is copied to index  $n = \text{limit}() - 1 - p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns:**

a reference to this **IntBuffer** (p. 2066)

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2062).

**6.404.3.6** **virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & value)**  
**const** [virtual]

**6.404.3.7** **virtual IntBuffer\* decaf::nio::IntBuffer::duplicate ()** [pure virtual]

Creates a new int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

a new int **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2062).

**6.404.3.8** **virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & value) const**  
[virtual]

**6.404.3.9** **IntBuffer& decaf::nio::IntBuffer::get (int \* buffer, int size,**  
**int offset, int length) throw ( BufferUnderflowException,**  
**decaf::lang::exceptions::IndexOutOfBoundsException,**  
**decaf::lang::exceptions::NullPointerException )**

Relative bulk get method. This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if  $\text{length} > \text{remaining}()$  (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies length ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

**Parameters:**

*buffer* The pointer to an allocated buffer to fill.  
*size* The size of the buffer that was passed in.  
*offset* The position in the buffer to start filling.  
*length* The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length ints remaining in this buffer.  
*NullPointerException* if the passed buffer is null.  
*IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

**6.404.3.10 IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > *buffer*) throw ( BufferUnderflowException )**

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize( N ) before calling this get method.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length ints remaining in this buffer.

**6.404.3.11 virtual int decaf::nio::IntBuffer::get (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]**

Absolute get method. Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the int is to be read.

**Returns:**

the int that is located at the given index.

**Exceptions:**

***IndexOutOfBoundsException*** if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2062).

**6.404.3.12** **virtual int decaf::nio::IntBuffer::get () throw ( BufferUnderflowException ) [pure virtual]**

Relative get method. Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the int at the current position.

**Exceptions:**

***BufferUnderflowException*** (p. 957) if there no more data to return.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2063).

**6.404.3.13** **virtual bool decaf::nio::IntBuffer::hasArray () const [pure virtual]**

Tells whether or not this buffer is backed by an accessible int array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2063).

**6.404.3.14** **virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & value) const [virtual]**

**6.404.3.15** **virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & value) const [virtual]**

**6.404.3.16** **virtual IntBuffer& decaf::nio::IntBuffer::put (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]**

Writes the given ints into this buffer at the given index.

**Parameters:**

***index*** The position in the **Buffer** (p. 928) to write the data.

***value*** The ints to write.

**Returns:**

a reference to this buffer.



**Exceptions:**

*IndexOutOfBoundsException* - If index greater than the buffer's limit minus the size of the type being written, or the index is negative.

*ReadOnlyBufferException* (p. 3169) - If this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2063).

**6.404.3.17 virtual IntBuffer& decaf::nio::IntBuffer::put (int *value*) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]**

Writes the given integer into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The integer value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2064).

**6.404.3.18 IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & *buffer*) throw ( BufferOverflowException, ReadOnlyBufferException )**

This method transfers the entire content of the given source ints array into this buffer. This is the same as calling `put( &buffer[0], 0, buffer.size()`.

**Parameters:**

*buffer* The buffer whose contents are copied to this **IntBuffer** (p. 2066).

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

**6.404.3.19** `IntBuffer& decaf::nio::IntBuffer::put (const int *  
buffer, int size, int offset, int length) throw (  
BufferOverflowException, ReadOnlyBufferException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

This method transfers ints into this buffer from the given source array. If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no ints are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters:**

*buffer* The array from which integers are to be read.

*size* The size of the buffer passed.

*offset* The offset within the array of the first char to be read.

*length* The number of integers to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**NullPointerException** if the passed buffer is null.

**IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

**6.404.3.20** `IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & src) throw  
( BufferOverflowException, ReadOnlyBufferException,  
decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the ints remaining in the given source buffer into this buffer. If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 933), then no ints are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

**Parameters:**

*src* The buffer to take ints from an place in this one.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer for the remaining ints in the source buffer.

*IllegalArgumentException* if the source buffer is this buffer.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

**6.404.3.21 virtual IntBuffer\* decaf::nio::IntBuffer::slice () const [pure virtual]**

Creates a new **IntBuffer** (p. 2066) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **IntBuffer** (p. 2066) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2064).

**6.404.3.22 virtual std::string decaf::nio::IntBuffer::toString () const [virtual]****Returns:**

a std::string describing this object

**6.404.3.23 static IntBuffer\* decaf::nio::IntBuffer::wrap (std::vector< int > & buffer) [static]**

Wraps the passed STL int Vector in a **IntBuffer** (p. 2066). The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).

**Returns:**

a new **IntBuffer** (p. 2066) that is backed by buffer, caller owns.

**6.404.3.24 static IntBuffer\* decaf::nio::IntBuffer::wrap (int \* array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]**

Wraps the passed buffer with a new **IntBuffer** (p. 2066). The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice

versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

- array*** The array that will back the new buffer.
- size*** The size of the passed in array.
- offset*** The offset of the subarray to be used.
- length*** The length of the subarray to be used.

**Returns:**

a new **IntBuffer** (p.2066) that is backed by buffer, caller owns.

**Exceptions:**

- NullPointerException*** if the array pointer is NULL.
- IndexOutOfBoundsException*** if the preconditions of size, offset, or length are not met.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/IntBuffer.h`

## 6.405 decaf::lang::Integer Class Reference

#include <src/main/decaf/lang/Integer.h> Inheritance diagram for decaf::lang::Integer:

### Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Integer** ()
- virtual int **compareTo** (const **Integer** &i) const  
*Compares this **Integer** (p. 2077) instance with another.*
- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Integer** &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const int &i) const  
*Compares this **Integer** (p. 2077) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const int &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static **Integer** **decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3665) into a **Integer** (p. 2077).*
- static int **reverseBytes** (int value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.*
- static int **reverse** (int value)  
*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.*
- static int **parseInt** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed int in the radix specified by the second argument.*
- static int **parseInt** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal int.*
- static **Integer** **valueOf** (int value)  
*Returns a **Integer** (p. 2077) instance representing the specified int value.*
- static **Integer** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Integer** (p. 2077) object holding the value given by the specified std::string.*
- static **Integer** **valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Integer** (p. 2077) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*
- static int **bitCount** (int value)  
*Returns the number of one-bits in the two's complement binary representation of the specified int value.*
- static std::string **toString** (int value)  
*Converts the int to a **String** (p. 3665) representation.*
- static std::string **toString** (int value, int radix)  
*Returns a string representation of the first argument in the radix specified by the second argument.*
- static std::string **toHexString** (int value)  
*Returns a string representation of the integer argument as an unsigned integer in base 16.*
- static std::string **toOctalString** (int value)  
*Returns a string representation of the integer argument as an unsigned integer in base 8.*
- static std::string **toBinaryString** (int value)  
*Returns a string representation of the integer argument as an unsigned integer in base 2.*

- static int **highestOneBit** (int value)  
*Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*
- static int **lowestOneBit** (int value)  
*Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*
- static int **numberOfLeadingZeros** (int value)  
*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.*
- static int **numberOfTrailingZeros** (int value)  
*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.*
- static int **rotateLeft** (int value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.*
- static int **rotateRight** (int value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.*
- static int **signum** (int value)  
*Returns the signum function of the specified int value.*

## Static Public Attributes

- static const int **SIZE** = 32  
*The size in bits of the primitive int type.*
- static const int **MAX\_VALUE** = (int)0x7FFFFFFF  
*The maximum value that the primitive type can hold.*
- static const int **MIN\_VALUE** = (int)0x80000000  
*The minimum value that the primitive type can hold.*

## 6.405.1 Constructor & Destructor Documentation

### 6.405.1.1 decaf::lang::Integer::Integer (int value)

#### Parameters:

*value* The primitive value to wrap in an Integer (p. 2077) instance.

**6.405.1.2** `decaf::lang::Integer::Integer (const std::string & value) throw ( exceptions::NumberFormatException )`

**Parameters:**

*value* The base 10 encoded string to decode to an `Integer` (p.2077) and wrap.

**Exceptions:**

*NumberFormatException*

**6.405.1.3** `virtual decaf::lang::Integer::~~Integer () [inline, virtual]`

## 6.405.2 Member Function Documentation

**6.405.2.1** `static int decaf::lang::Integer::bitCount (int value) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

**Parameters:**

*value* - the int to count

**Returns:**

the number of one-bits in the two's complement binary representation of the specified int value.

**6.405.2.2** `virtual unsigned char decaf::lang::Integer::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p.2835).

**6.405.2.3** `virtual int decaf::lang::Integer::compareTo (const int & i) const [virtual]`

Compares this `Integer` (p.2077) instance with another.

**Parameters:**

*i* - the `Integer` (p.2077) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< int >` (p.1214).



**6.405.2.4 virtual int decaf::lang::Integer::compareTo (const Integer & i) const**  
[virtual]

Compares this **Integer** (p.2077) instance with another.

**Parameters:**

*i* - the **Integer** (p.2077) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.405.2.5 static Integer decaf::lang::Integer::decode (const std::string & value)**  
**throw ( exceptions::NumberFormatException )** [static]

Decodes a **String** (p.3665) into a **Integer** (p.2077). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p.3665) is the minus sign. No whitespace characters are permitted in the string.

**Parameters:**

*value* - The string to decode

**Returns:**

a **Integer** (p.2077) object containing the decoded value

**Exceptions:**

**NumberFormatException** if the string is not formatted correctly.

**6.405.2.6 virtual double decaf::lang::Integer::doubleValue () const** [inline, virtual]

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p.2836).

**6.405.2.7 bool decaf::lang::Integer::equals (const int & i) const** [inline, virtual]**Parameters:**

*i* - the **Integer** (p.2077) object to compare against.

**Returns:**

true if the two **Integer** (p. 2077) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p. 1215).

**6.405.2.8    bool decaf::lang::Integer::equals (const Integer & i) const    [inline]****Parameters:**

*i* - the **Integer** (p. 2077) object to compare against.

**Returns:**

true if the two **Integer** (p. 2077) Objects have the same value.

**6.405.2.9    virtual float decaf::lang::Integer::floatValue () const    [inline, virtual]**

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.405.2.10    static int decaf::lang::Integer::highestOneBit (int value)    [static]**

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters:**

*value* - the int to be inspected

**Returns:**

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.405.2.11    virtual int decaf::lang::Integer::intValue () const    [inline, virtual]**

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.405.2.12** `virtual long long decaf::lang::Integer::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.405.2.13** `static int decaf::lang::Integer::lowestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters:**

*value* - the int to be inspected

**Returns:**

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.405.2.14** `static int decaf::lang::Integer::numberOfLeadingZeros (int value) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\ast \text{floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x) \ast \text{ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

**Parameters:**

*value* - the int to be inspected

**Returns:**

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

**6.405.2.15** `static int decaf::lang::Integer::numberOfTrailingZeros (int value) [static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

**Parameters:**

*value* - the int to be inspected

**Returns:**

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

**6.405.2.16**    **virtual bool decaf::lang::Integer::operator< (const int & i) const**  
                 [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*i* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 1215).

**6.405.2.17**    **virtual bool decaf::lang::Integer::operator< (const Integer & i) const**  
                 [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*i* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.405.2.18**    **virtual bool decaf::lang::Integer::operator== (const int & i) const**  
                 [inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*i* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 1215).

**6.405.2.19** `virtual bool decaf::lang::Integer::operator==(const Integer & i) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*i* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.405.2.20** `static int decaf::lang::Integer::parseInt (const std::string & s) throw (`  
`exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed decimal int. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseInt( const std::string, int ) method.

**Parameters:**

*s* - **String** (p. 3665) to convert to a int

**Returns:**

the converted int value

**Exceptions:**

*NumberFormatException* if the string is not a int.

**6.405.2.21** `static int decaf::lang::Integer::parseInt (const std::string & s, int radix)`  
`throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed int in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1103) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type *NumberFormatException* is thrown if any of the following situations occurs:  
\* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1107) or larger than **Character.MAX\_RADIX** (p. 1106). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type int.

**Parameters:**

*s* - the **String** (p. 3665) containing the int representation to be parsed

*radix* - the radix to be used while parsing s

**Returns:**

the int represented by the string argument in the specified radix.

**Exceptions:**

*NumberFormatException* - If **String** (p. 3665) does not contain a parsable int.

**6.405.2.22 static int decaf::lang::Integer::reverse (int *value*) [static]**

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

**Parameters:**

*value* - the value whose bits are to be reversed

**Returns:**

the reversed bits int.

**6.405.2.23 static int decaf::lang::Integer::reverseBytes (int *value*) [static]**

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

**Parameters:**

*value* - the int whose bytes we are to reverse

**Returns:**

the reversed int.

**6.405.2.24 static int decaf::lang::Integer::rotateLeft (int *value*, int *distance*) [static]**

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

**Parameters:**

*value* - the int to be inspected

*distance* - the number of bits to rotate

**Returns:**

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

**6.405.2.25 static int decaf::lang::Integer::rotateRight (int *value*, int *distance*)**  
[static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

**Parameters:**

*value* - the int to be inspected

*distance* - the number of bits to rotate

**Returns:**

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

**6.405.2.26 virtual short decaf::lang::Integer::shortValue () const** [inline, virtual]

Answers the short value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2837).

**6.405.2.27 static int decaf::lang::Integer::signum (int *value*)** [static]

Returns the signum function of the specified int value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

**Parameters:**

*value* - the int to be inspected

**Returns:**

the signum function of the specified int value.

**6.405.2.28 static std::string decaf::lang::Integer::toBinaryString (int *value*)**  
[static]

Returns a string representation of the integer argument as an unsigned integer in base 2. The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

**Parameters:**

*value* - the int to be translated to a binary string

**Returns:**

the unsigned int value as a binary string

**6.405.2.29 static std::string decaf::lang::Integer::toHexString (int *value*) [static]**

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

**Parameters:**

*value* - the int to be translated to an Octal string

**Returns:**

the unsigned int value as a Octal string

**6.405.2.30 static std::string decaf::lang::Integer::toOctalString (int *value*) [static]**

Returns a string representation of the integer argument as an unsigned integer in base 8. The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

**Parameters:**

*value* - the int to be translated to an Octal string

**Returns:**

the unsigned int value as a Octal string

**6.405.2.31 static std::string decaf::lang::Integer::toString (int *value*, int *radix*) [static]**

Returns a string representation of the first argument in the radix specified by the second argument. If the radix is smaller than **Character.MIN\_RADIX** (p.1107) or larger than **Character.MAX\_RADIX** (p.1106), then the radix 10 is used instead.



If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

**Parameters:**

*value* - the int to convert to a string

*radix* - the radix to format the string in

**Returns:**

an int formatted to the string value of the radix given.

**6.405.2.32** static std::string decaf::lang::Integer::toString (int *value*) [static]

Converts the int to a **String** (p. 3665) representation.

**Parameters:**

*value* The int to convert to a std::string instance.

**Returns:**

string representation

**6.405.2.33** std::string decaf::lang::Integer::toString () const

**Returns:**

this Integer (p. 2077) Object as a **String** (p. 3665) Representation

**6.405.2.34** static Integer decaf::lang::Integer::valueOf (const std::string & *value*, int *radix*) throw ( exceptions::NumberFormatException ) [static]

Returns a **Integer** (p. 2077) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the parseInt( std::string, int ) method. The result is a **Integer** (p. 2077) object that represents the int value specified by the string.

**Parameters:**

*value* - std::string to parse as base ( radix )

*radix* - base of the string to parse.

**Returns:**

new **Integer** (p. 2077) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a valid int.

**6.405.2.35**    `static Integer decaf::lang::Integer::valueOf (const std::string & value)  
                 throw ( exceptions::NumberFormatException )    [static]`

Returns a **Integer** (p. 2077) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt( std::string )` method. The result is a **Integer** (p. 2077) object that represents the int value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base 10

**Returns:**

new **Integer** (p. 2077) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a decimal int.

**6.405.2.36**    `static Integer decaf::lang::Integer::valueOf (int value)    [inline, static]`

Returns a **Integer** (p. 2077) instance representing the specified int value.

**Parameters:**

*value* - the int to wrap

**Returns:**

the new **Integer** (p. 2077) object wrapping value.

### 6.405.3    Field Documentation

**6.405.3.1**    `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF  
                 [static]`

The maximum value that the primitive type can hold.

**6.405.3.2**    `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000    [static]`

The minimum value that the primitive type can hold.

**6.405.3.3**    `const int decaf::lang::Integer::SIZE = 32    [static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

## 6.406 activemq::commands::IntegerResponse Class Reference

#include <src/main/activemq/commands/IntegerResponse.h> Inheritance diagram for activemq::commands::IntegerResponse:

### Public Member Functions

- **IntegerResponse** ()
- virtual **~IntegerResponse** ()
- virtual unsigned char **getDataStructureType** () const

*Get the unique identifier that this object and its own Marshaler share.*

- virtual **IntegerResponse \* cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*

- virtual int **getResult** () const
- virtual void **setResult** (int result)

### Static Public Attributes

- static const unsigned char **ID\_INTEGERRESPONSE** = 34

### Protected Attributes

- int **result**

## 6.406.1 Constructor & Destructor Documentation

**6.406.1.1** `activemq::commands::IntegerResponse::IntegerResponse ()`

**6.406.1.2** `virtual activemq::commands::IntegerResponse::~~IntegerResponse ()`  
[virtual]

## 6.406.2 Member Function Documentation

**6.406.2.1** `virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.406.2.2** `virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.406.2.3** `virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 3286).

**6.406.2.4** `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Reimplemented from `activemq::commands::Response` (p.3287).

**6.406.2.5** `virtual int activemq::commands::IntegerResponse::getResult () const`  
[virtual]

**6.406.2.6** `virtual void activemq::commands::IntegerResponse::setResult (int result)`  
[virtual]

**6.406.2.7** `virtual std::string activemq::commands::IntegerResponse::toString ()`  
`const` [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::Response` (p.3287).

### 6.406.3 Field Documentation

**6.406.3.1** `const unsigned char activemq::commands::IntegerResponse::ID_-`  
`INTEGERRESPONSE = 34` [static]

**6.406.3.2** `int activemq::commands::IntegerResponse::result` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/IntegerResponse.h`

## 6.407 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2094).

#include <src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.407.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2094).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.407.2 Constructor & Destructor Documentation

**6.407.2.1** `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.407.2.2** `virtual activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.407.3 Member Function Documentation

**6.407.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.407.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.407.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3321).

**6.407.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).

**6.407.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3322).



**6.407.3.6** virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p.3323).

**6.407.3.7** virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p.3324).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**IntegerResponseMarshaller.h**

## 6.408 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2098).

#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.408.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2098).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.408.2 Constructor & Destructor Documentation

**6.408.2.1** `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.408.2.2** `virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.408.3 Member Function Documentation

**6.408.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.408.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.408.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3301).

**6.408.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.408.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3302).

**6.408.3.6** virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3303).

**6.408.3.7** virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3304).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**

## 6.409 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2102).

#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.409.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2102).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.409.2 Constructor & Destructor Documentation

**6.409.2.1** `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.409.2.2** `virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.409.3 Member Function Documentation

**6.409.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.409.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.409.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3311).

**6.409.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).

**6.409.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3312).



**6.409.3.6** virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p.3313).

**6.409.3.7** virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p.3314).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**

## 6.410 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2106).

#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.410.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2106).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.410.2 Constructor & Destructor Documentation

**6.410.2.1** `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.410.2.2** `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.410.3 Member Function Documentation

**6.410.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.410.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.410.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3296).

**6.410.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.410.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3297).

**6.410.3.6** virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3298).

**6.410.3.7** virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3299).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**IntegerResponseMarshaller.h**

## 6.411 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2110).

#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.411.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2110).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.411.2 Constructor & Destructor Documentation

**6.411.2.1** `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.411.2.2** `virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.411.3 Member Function Documentation

**6.411.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.411.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.411.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3316).

**6.411.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).

**6.411.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3317).



**6.411.3.6** virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3318).

**6.411.3.7** virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**

## 6.412 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2114).

#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.412.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2114).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.412.2 Constructor & Destructor Documentation

**6.412.2.1** `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

**6.412.2.2** `virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

## 6.412.3 Member Function Documentation

**6.412.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.412.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.412.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3306).

**6.412.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.412.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3307).

**6.412.3.6** virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p.3308).

**6.412.3.7** virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p.3309).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**IntegerResponseMarshaller.h**

## 6.413 internal\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- **z\_stream** strm
- **int** status
- **Bytef** \* pending\_buf
- **ulg** pending\_buf\_size
- **Bytef** \* pending\_out
- **uInt** pending
- **int** wrap
- **gz\_headerp** gzhead
- **uInt** gzindex
- **Byte** method
- **int** last\_flush
- **uInt** w\_size
- **uInt** w\_bits
- **uInt** w\_mask
- **Bytef** \* window
- **ulg** window\_size
- **Posf** \* prev
- **Posf** \* head
- **uInt** ins\_h
- **uInt** hash\_size
- **uInt** hash\_bits
- **uInt** hash\_mask
- **uInt** hash\_shift
- **long** block\_start
- **uInt** match\_length
- **IPos** prev\_match
- **int** match\_available
- **uInt** strstart
- **uInt** match\_start
- **uInt** lookahead
- **uInt** prev\_length
- **uInt** max\_chain\_length
- **uInt** max\_lazy\_match
- **int** level
- **int** strategy
- **uInt** good\_match
- **int** nice\_match
- **struct ct\_data\_s** dyn\_ltree [HEAP\_SIZE]
- **struct ct\_data\_s** dyn\_dtree [2 \*D\_CODES+1]
- **struct ct\_data\_s** bl\_tree [2 \*BL\_CODES+1]
- **struct tree\_desc\_s** l\_desc
- **struct tree\_desc\_s** d\_desc
- **struct tree\_desc\_s** bl\_desc
- **ush** bl\_count [MAX\_BITS+1]

- `int heap [2 * L_CODES + 1]`
- `int heap_len`
- `int heap_max`
- `uch depth [2 * L_CODES + 1]`
- `uchf * l_buf`
- `uInt lit_bufsize`
- `uInt last_lit`
- `ushf * d_buf`
- `ulg opt_len`
- `ulg static_len`
- `uInt matches`
- `int last_eob_len`
- `ush bi_buf`
- `int bi_valid`
- `ulg high_water`
- `int dummy`

### 6.413.1 Field Documentation

- 6.413.1.1 ush internal\_state::bi\_buf
- 6.413.1.2 int internal\_state::bi\_valid
- 6.413.1.3 ush internal\_state::bl\_count[MAX\_BITS+1]
- 6.413.1.4 struct tree\_desc\_s internal\_state::bl\_desc [read]
- 6.413.1.5 struct ct\_data\_s internal\_state::bl\_tree[2 \*BL\_CODES+1] [read]
- 6.413.1.6 long internal\_state::block\_start
- 6.413.1.7 ushf\* internal\_state::d\_buf
- 6.413.1.8 struct tree\_desc\_s internal\_state::d\_desc [read]
- 6.413.1.9 uch internal\_state::depth[2 \*L\_CODES+1]
- 6.413.1.10 int internal\_state::dummy
- 6.413.1.11 struct ct\_data\_s internal\_state::dyn\_dtree[2 \*D\_CODES+1] [read]
- 6.413.1.12 struct ct\_data\_s internal\_state::dyn\_ltree[HEAP\_SIZE] [read]
- 6.413.1.13 uInt internal\_state::good\_match
- 6.413.1.14 gz\_headerp internal\_state::gzhead
- 6.413.1.15 uInt internal\_state::gzindex
- 6.413.1.16 uInt internal\_state::hash\_bits
- 6.413.1.17 uInt internal\_state::hash\_mask
- 6.413.1.18 uInt internal\_state::hash\_shift
- 6.413.1.19 uInt internal\_state::hash\_size
- 6.413.1.20 Posf\* internal\_state::head
- 6.413.1.21 int internal\_state::heap[2 \*L\_CODES+1]
- 6.413.1.22 int internal\_state::heap\_len
- 6.413.1.23 int internal\_state::heap\_max
- 6.413.1.24 ulg internal\_state::high\_water
- 6.413.1.25 uInt internal\_state::ins\_h
- 6.413.1.26 uchf\* internal\_state::l\_buf
- 6.413.1.27 struct tree\_desc\_s internal\_state::l\_desc [read]
- 6.413.1.28 int internal\_state::last\_eob\_len
- 6.413.1.29 int internal\_state::last\_flush
- 6.413.1.30 uInt internal\_state::last\_lit



- src/main/decaf/internal/util/zip/**deflate.h**
- src/main/decaf/internal/util/zip/**zlib.h**

## 6.414 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2768).

#include <src/main/activemq/transport/mock/InternalCommandListener.h>Inheritance diagram for activemq::transport::mock::InternalCommandListener:

### Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** \*transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- void **run** ()  
*Default implementation of the run method - does nothing.*

### 6.414.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2768). This class processes all outbound **commands** (p. 87) and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 3289) and getting a set of Commands to send back into the **MockTransport** (p. 2768) as incoming Commands and Responses.

### 6.414.2 Constructor & Destructor Documentation

**6.414.2.1** **activemq::transport::mock::InternalCommandListener::InternalCommandListener** ()

**6.414.2.2** **virtual**  
**activemq::transport::mock::InternalCommandListener::~~InternalCommandListener**  
 () [virtual]

### 6.414.3 Member Function Documentation

**6.414.3.1** **virtual void** **activemq::transport::mock::InternalCommandListener::onCommand** (const **Pointer**< **Command** > & *command*) [virtual]

Event handler for the receipt of a command. The **transport** (p. 97) passes off all received **commands** (p. 87) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3883) deletes the command upon receipt.

**Parameters:**

*command* the received command object.

Implements **activemq::transport::TransportListener** (p. 3900).

**6.414.3.2** `void activemq::transport::mock::InternalCommandListener::run ()`  
[virtual]

Default implementation of the run method - does nothing.

Reimplemented from **decaf::lang::Thread** (p. 3771).

**6.414.3.3** `void activemq::transport::mock::InternalCommandListener::setResponseBuilder`  
(const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

**6.414.3.4** `void activemq::transport::mock::InternalCommandListener::setTransport`  
(MockTransport \* *transport*) [inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/InternalCommandListener.h`

## 6.415 decaf::lang::exceptions::InterruptedException Class Reference

#include <src/main/decaf/lang/exceptions/InterruptedException.h> Inheritance diagram for decaf::lang::exceptions::InterruptedException:

### Public Member Functions

- **InterruptedException** () throw ()  
*Default Constructor.*
- **InterruptedException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()  
*Copy Constructor.*
- **InterruptedException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InterruptedException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InterruptedException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InterruptedException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InterruptedException** () throw ()

### 6.415.1 Constructor & Destructor Documentation

#### 6.415.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

#### 6.415.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.415.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException**  
**(const InterruptedException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p.1831) whose data is to be copied into this one.

**6.415.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException**  
**(const char \* *file*, const int *lineNumber*, const std::exception \* *cause*,  
const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.415.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException**  
**(const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p.2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.415.1.6 decaf::lang::exceptions::InterruptedException::InterruptedException**  
**(const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()**  
**[inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.415.1.7**    **virtual**  
          `decaf::lang::exceptions::InterruptedException::~~InterruptedException ()`  
          `throw ()`    [inline, virtual]

## **6.415.2    Member Function Documentation**

**6.415.2.1**    **virtual** `InterruptedException*` `de-`  
          `caf::lang::exceptions::InterruptedException::clone ()` **const**  
          [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InterruptedException.h`

## 6.416 decaf::io::InterruptedIOException Class Reference

#include <src/main/decaf/io/InterruptedIOException.h> Inheritance diagram for decaf::io::InterruptedIOException:

### Public Member Functions

- **InterruptedIOException** () throw ()  
*Default Constructor.*
- **InterruptedIOException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **InterruptedIOException** (const InterruptedIOException &ex) throw ()  
*Copy Constructor.*
- **InterruptedIOException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InterruptedIOException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InterruptedIOException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **InterruptedIOException** \* clone () const  
*Clones this exception.*
- virtual ~**InterruptedIOException** () throw ()

### 6.416.1 Constructor & Destructor Documentation

#### 6.416.1.1 decaf::io::InterruptedIOException::InterruptedIOException () throw () [inline]

Default Constructor.

#### 6.416.1.2 decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

#### 6.416.1.3 `decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & ex) throw () [inline]`

Copy Constructor.

##### Parameters:

*ex* the exception to copy, which is an instance of this type

#### 6.416.1.4 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

#### 6.416.1.5 `decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * cause) throw () [inline]`

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.416.1.6 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

##### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message



**6.416.1.7** virtual decaf::io::InterruptedIOException::~~InterruptedIOException ()  
throw () [inline, virtual]

## 6.416.2 Member Function Documentation

**6.416.2.1** virtual InterruptedIOException\* decaf::io::InterruptedIOException::clone  
( ) const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2144).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 3544).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

## 6.417 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

#include <src/main/cms/InvalidClientIdException.h> Inheritance diagram for cms::InvalidClientIdException:

### Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

### 6.417.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since:

1.3

### 6.417.2 Constructor & Destructor Documentation

- 6.417.2.1** cms::InvalidClientIdException::InvalidClientIdException () throw ()
- 6.417.2.2** cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & ex) throw ()
- 6.417.2.3** cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception \* cause) throw ()
- 6.417.2.4** cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()
- 6.417.2.5** virtual cms::InvalidClientIdException::~InvalidClientIdException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

## 6.418 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

#include <src/main/cms/InvalidDestinationException.h> Inheritance diagram for cms::InvalidDestinationException:

### Public Member Functions

- **InvalidDestinationException** () throw ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidDestinationException** () throw ()

### 6.418.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since:

1.3

### 6.418.2 Constructor & Destructor Documentation

- 6.418.2.1 **cms::InvalidDestinationException::InvalidDestinationException** () throw ()
- 6.418.2.2 **cms::InvalidDestinationException::InvalidDestinationException** (const **InvalidDestinationException** & *ex*) throw ()
- 6.418.2.3 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & *message*, const std::exception \* *cause*) throw ()
- 6.418.2.4 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & *message*, const std::exception \* *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.418.2.5 virtual **cms::InvalidDestinationException::~~InvalidDestinationException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidDestinationException.h**

## 6.419 decaf::security::InvalidKeyException Class Reference

#include <src/main/decaf/security/InvalidKeyException.h> Inheritance diagram for decaf::security::InvalidKeyException:

### Public Member Functions

- **InvalidKeyException** () throw ()  
*Default Constructor.*
- **InvalidKeyException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()  
*Copy Constructor.*
- **InvalidKeyException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidKeyException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidKeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidKeyException** \* clone () const  
*Clones this exception.*
- virtual ~**InvalidKeyException** () throw ()

### 6.419.1 Constructor & Destructor Documentation

#### 6.419.1.1 decaf::security::InvalidKeyException::InvalidKeyException () throw () [inline]

Default Constructor.

#### 6.419.1.2 decaf::security::InvalidKeyException::InvalidKeyException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.419.1.3 decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.419.1.4 decaf::security::InvalidKeyException::InvalidKeyException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.419.1.5 decaf::security::InvalidKeyException::InvalidKeyException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.419.1.6 decaf::security::InvalidKeyException::InvalidKeyException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.419.1.7** `virtual decaf::security::InvalidKeyException::~~InvalidKeyException ()  
throw () [inline, virtual]`

## **6.419.2 Member Function Documentation**

**6.419.2.1** `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone  
() const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

A deep copy of this exception.

Reimplemented from `decaf::security::KeyException` (p. 2297).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

## 6.420 decaf::nio::InvalidMarkException Class Reference

#include <src/main/decaf/nio/InvalidMarkException.h> Inheritance diagram for decaf::nio::InvalidMarkException:

### Public Member Functions

- **InvalidMarkException** () throw ()  
*Default Constructor.*
- **InvalidMarkException** (const lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **InvalidMarkException** (const InvalidMarkException &ex) throw ()  
*Copy Constructor.*
- **InvalidMarkException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidMarkException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidMarkException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidMarkException** \* clone () const  
*Clones this exception.*
- virtual ~**InvalidMarkException** () throw ()

### 6.420.1 Constructor & Destructor Documentation

#### 6.420.1.1 decaf::nio::InvalidMarkException::InvalidMarkException () throw () [inline]

Default Constructor.

#### 6.420.1.2 decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* The Exception whose state data is to be copied into this Exception.

#### 6.420.1.3 `decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & ex) throw () [inline]`

Copy Constructor.

##### Parameters:

*ex* The Exception whose state data is to be copied into this Exception.

#### 6.420.1.4 `decaf::nio::InvalidMarkException::InvalidMarkException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

#### 6.420.1.5 `decaf::nio::InvalidMarkException::InvalidMarkException (const std::exception * cause) throw () [inline]`

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.420.1.6 `decaf::nio::InvalidMarkException::InvalidMarkException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message



**6.420.1.7** virtual decaf::nio::InvalidMarkException::~InvalidMarkException ()  
throw () [inline, virtual]

## 6.420.2 Member Function Documentation

**6.420.2.1** virtual InvalidMarkException\* decaf::nio::InvalidMarkException::clone ()  
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A new Exception instance that is a copy of this Exception.

Reimplemented from **decaf::lang::exceptions::IllegalStateException** (p. 2000).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**InvalidMarkException.h**

## 6.421 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

#include <src/main/cms/InvalidSelectorException.h> Inheritance diagram for cms::InvalidSelectorException:

### Public Member Functions

- **InvalidSelectorException** () throw ()
- **InvalidSelectorException** (const **InvalidSelectorException** &ex) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidSelectorException** () throw ()

#### 6.421.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since:

1.3

#### 6.421.2 Constructor & Destructor Documentation

**6.421.2.1 cms::InvalidSelectorException::InvalidSelectorException () throw ()**

**6.421.2.2 cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & ex) throw ()**

**6.421.2.3 cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception \* cause) throw ()**

**6.421.2.4 cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()**

**6.421.2.5 virtual cms::InvalidSelectorException::~InvalidSelectorException () throw () [virtual]**

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidSelectorException.h**

## 6.422 decaf::lang::exceptions::InvalidStateException Class Reference

#include <src/main/decaf/lang/exceptions/InvalidStateException.h> Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

### Public Member Functions

- **InvalidStateException** () throw ()  
*Default Constructor.*
- **InvalidStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()  
*Copy Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidStateException** \* clone () const  
*Clones this exception.*
- virtual ~**InvalidStateException** () throw ()

### 6.422.1 Constructor & Destructor Documentation

#### 6.422.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw () [inline]

Default Constructor.

#### 6.422.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

### 6.422.1.3 `decaf::lang::exceptions::InvalidStateException::InvalidStateException` `(const InvalidStateException & ex) throw () [inline]`

Copy Constructor.

#### Parameters:

*ex* The **Exception** (p.1831) whose data is to be copied into this one.

### 6.422.1.4 `decaf::lang::exceptions::InvalidStateException::InvalidStateException` `(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.422.1.5 `decaf::lang::exceptions::InvalidStateException::InvalidStateException` `(const std::exception * cause) throw () [inline]`

Constructor.

#### Parameters:

*cause* **Pointer** (p.2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.422.1.6 `decaf::lang::exceptions::InvalidStateException::InvalidStateException` `(const char * file, const int lineNumber, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.422.1.7**    **virtual**  
          **decaf::lang::exceptions::InvalidStateException::~~InvalidStateException ()**  
          **throw ()**    [inline, virtual]

## 6.422.2 Member Function Documentation

**6.422.2.1**    **virtual InvalidStateException\* de-**  
          **caf::lang::exceptions::InvalidStateException::clone () const**  
          [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

## 6.423 decaf::io::IOException Class Reference

#include <src/main/decaf/io/IOException.h> Inheritance diagram for decaf::io::IOException:

### Public Member Functions

- **IOException** () throw ()  
*Default Constructor.*
- **IOException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **IOException** (const IOException &ex) throw ()  
*Copy Constructor.*
- **IOException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IOException** (const std::exception \*cause) throw ()  
*Constructor.*
- **IOException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **IOException** \* clone () const  
*Clones this exception.*
- virtual ~**IOException** () throw ()

### 6.423.1 Constructor & Destructor Documentation

#### 6.423.1.1 decaf::io::IOException::IOException () throw () [inline]

Default Constructor.

#### 6.423.1.2 decaf::io::IOException::IOException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

**6.423.1.3 decaf::io::IOException::IOException (const IOException & *ex*) throw ()**  
[inline]

Copy Constructor.

**Parameters:**

*ex* the exception to copy, which is an instance of this type

**6.423.1.4 decaf::io::IOException::IOException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw ()**  
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.423.1.5 decaf::io::IOException::IOException (const std::exception \* *cause*) throw ()**  
[inline]

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.423.1.6 decaf::io::IOException::IOException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()**  
[inline]

Constructor.

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.423.1.7** `virtual decaf::io::IOException::~~IOException () throw () [inline, virtual]`

## 6.423.2 Member Function Documentation

**6.423.2.1** `virtual IOException* decaf::io::IOException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new instance of an Exception that is a copy of this instance.

Reimplemented from `decaf::lang::Exception` (p. 1834).

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketException` (p. 2873), `decaf::io::EOFException` (p. 1827), `decaf::io::InterruptedIOException` (p. 2129), `decaf::io::UnsupportedEncodingException` (p. 3915), `decaf::io::UTFDataFormatException` (p. 3968), `decaf::net::BindException` (p. 838), `decaf::net::ConnectException` (p. 1261), `decaf::net::HttpRetryException` (p. 1988), `decaf::net::MalformedURLException` (p. 2458), `decaf::net::NoRouteToHostException` (p. 2822), `decaf::net::PortUnreachableException` (p. 2975), `decaf::net::ProtocolException` (p. 3139), `decaf::net::SocketException` (p. 3522), `decaf::net::SocketTimeoutException` (p. 3544), `decaf::net::UnknownHostException` (p. 3909), `decaf::net::UnknownServiceException` (p. 3912), and `decaf::util::zip::ZipException` (p. 4066).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`



## 6.424 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3883) interface that performs marshaling of **commands** (p. 87) to IO streams.

#include <src/main/activemq/transport/IOTransport.h> Inheritance diagram for activemq::transport::IOTransport:

### Public Member Functions

- **IOTransport** ()  
*Default Constructor.*
- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)  
*Create an instance of this **Transport** (p. 3883) and assign its **WireFormat** instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Not supported by this class - throws an exception.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Not supported by this class - throws an exception.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)  
*Sets the **WireFormat** instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual void **setInputStream** (decaf::io::DataInputStream \*is)  
*Sets the input stream for in-coming **commands** (p. 87).*
- virtual void **setOutputStream** (decaf::io::DataOutputStream \*os)  
*Sets the output stream for out-going **commands** (p. 87).*
- virtual void **start** () throw ( decaf::io::IOException )

Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87).

- virtual void **stop** () throw ( decaf::io::IOException )

*Stops the **Transport** (p. 3883), terminating any **threads** (p. 96) and stopping all read and write operations.*

- virtual void **close** () throw ( decaf::io::IOException )

*Stops the polling thread and closes the streams.*

- virtual void **run** ()

*Runs the polling thread.*

- virtual **Transport \* narrow** (const std::type\_info &typeId)

*Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3883) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3883) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const

- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*reconnect to another location*

### 6.424.1 Detailed Description

Implementation of the **Transport** (p. 3883) interface that performs marshaling of **commands** (p. 87) to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming **commands** (p. 87). When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### 6.424.2 Constructor & Destructor Documentation

#### 6.424.2.1 activemq::transport::IOTransport::IOTransport ()

Default Constructor.

### 6.424.2.2 `activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & wireFormat)`

Create an instance of this **Transport** (p. 3883) and assign its **WireFormat** instance at creation time.

#### Parameters:

*wireFormat* Data encoder / decoder to use when reading and writing.

### 6.424.2.3 `virtual activemq::transport::IOTransport::~~IOTransport ()` [virtual]

## 6.424.3 Member Function Documentation

### 6.424.3.1 `virtual void activemq::transport::IOTransport::close () throw ( decaf::io::IOException )` [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions:

*IOException* if errors occur.

Implements **decaf::io::Closeable** (p. 1149).

### 6.424.3.2 `virtual std::string activemq::transport::IOTransport::getRemoteAddress () const` [inline, virtual]

#### Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3884).

### 6.424.3.3 `virtual TransportListener* activemq::transport::IOTransport::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).

#### Returns:

The listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3884).

### 6.424.3.4 `virtual bool activemq::transport::IOTransport::isClosed () const` [inline, virtual]

Has the **Transport** (p. 3883) been shutdown and no longer usable.

**Returns:**

true if the **Transport** (p. 3883)

Implements **activemq::transport::Transport** (p. 3885).

**6.424.3.5 virtual bool activemq::transport::IOTransport::isConnected () const**  
[inline, virtual]

Is the **Transport** (p. 3883) Connected to its Broker.

**Returns:**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3885).

**6.424.3.6 virtual bool activemq::transport::IOTransport::isFaultTolerant () const**  
[inline, virtual]

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns:**

true if the **Transport** (p. 3883) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3885).

**6.424.3.7 virtual Transport\* activemq::transport::IOTransport::narrow (const**  
**std::type\_info & typeId) [inline, virtual]**

Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.

**Parameters:**

*typeId* - The type\_info of the Object we are searching for.

**Returns:**

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3885).

**6.424.3.8 virtual void activemq::transport::IOTransport::oneway (const**  
**Pointer< Command > & command) throw ( decaf::io::IOException,**  
**decaf::lang::exceptions::UnsupportedOperationException ) [virtual]**

Sends a one-way command. Does not wait for any response from the broker.

**Parameters:**

*command* the command to be sent.

**Exceptions:**

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3886).

**6.424.3.9** `virtual void activemq::transport::IOTransport::reconnect  
(const decaf::net::URI &uri AMQCPP_UNUSED) throw (  
decaf::io::IOException ) [inline, virtual]`

reconnect to another location

**Parameters:**

*uri*

**Exceptions:**

*IOException* on failure of if not supported

**6.424.3.10** `virtual Pointer<Response> activemq::transport::IOTransport::request  
(const Pointer< Command > & command, unsigned  
int timeout) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

**Parameters:**

*command* the command to be sent.

*timeout* the time to wait for a response.

**Returns:**

the response to the command sent.

**Exceptions:**

*UnsupportedOperationException*.

Implements **activemq::transport::Transport** (p. 3886).

**6.424.3.11** `virtual Pointer<Response> activemq::transport::IOTransport::request  
(const Pointer< Command > & command)  
throw ( decaf::io::IOException, de-  
caaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Not supported by this class - throws an exception.

**Parameters:**

*command* the command to be sent.

**Returns:**

the response to the command sent.

**Exceptions:**

*UnsupportedOperationException.*

Implements **activemq::transport::Transport** (p. 3887).

**6.424.3.12** **virtual void activemq::transport::IOTransport::run ()** [virtual]

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 3325).

**6.424.3.13** **virtual void activemq::transport::IOTransport::setInputStream**  
(**decaf::io::DataInputStream \* is**) [inline, virtual]

Sets the input stream for in-coming **commands** (p. 87).

**Parameters:**

*is* The input stream.

**6.424.3.14** **virtual void activemq::transport::IOTransport::setOutputStream**  
(**decaf::io::DataOutputStream \* os**) [inline, virtual]

Sets the output stream for out-going **commands** (p. 87).

**Parameters:**

*os* The output stream.

**6.424.3.15** **virtual void activemq::transport::IOTransport::setTransportListener**  
(**TransportListener \* listener**) [inline, virtual]

Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).

**Parameters:**

*listener* the listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3887).

**6.424.3.16** **virtual void activemq::transport::IOTransport::setWireFormat** (**const**  
**Pointer< wireformat::WireFormat > & wireFormat**) [inline, virtual]

Sets the WireFormat instance to use.

**Parameters:**

*wireFormat* The WireFormat the object used to encode / decode **commands** (p. 87).

Implements **activemq::transport::Transport** (p. 3888).

**6.424.3.17** virtual void activemq::transport::IOTransport::start () throw ( decaf::io::IOException ) [virtual]

Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions:**

*CMSEException* if an error occurs or if this **transport** (p. 97) has already been closed.

Implements **activemq::transport::Transport** (p. 3888).

**6.424.3.18** virtual void activemq::transport::IOTransport::stop () throw ( decaf::io::IOException ) [virtual]

Stops the **Transport** (p. 3883), terminating any **threads** (p. 96) and stopping all read and write operations.

**Exceptions:**

*IOException* if an error occurs while stopping the **Transport** (p. 3883).

Implements **activemq::transport::Transport** (p. 3888).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**IOTransport.h**

## 6.425 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 2152) type for generic collections API calls.

#include <src/main/decaf/lang/Iterable.h> Inheritance diagram for decaf::lang::Iterable< E >:

### Public Member Functions

- virtual `~Iterable ()`
- virtual `decaf::util::Iterator< E > * iterator ()=0`
- virtual `decaf::util::Iterator< E > * iterator () const =0`

#### 6.425.1 Detailed Description

`template<typename E> class decaf::lang::Iterable< E >`

Implementing this interface allows an object to be cast to an **Iterable** (p. 2152) type for generic collections API calls.

#### 6.425.2 Constructor & Destructor Documentation

**6.425.2.1** `template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable ()`  
[inline, virtual]

#### 6.425.3 Member Function Documentation

**6.425.3.1** `template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator () const` [pure virtual]

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3721), `decaf::util::PriorityQueue< E >` (p. 3032), `decaf::util::StlList< E >` (p. 3598), `decaf::util::StlSet< E >` (p. 3627), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), `decaf::util::StlList< cms::Connection * >` (p. 3598), `decaf::util::StlSet< transport::TransportListener * >` (p. 3627), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3627), `decaf::util::StlSet< Resource * >` (p. 3627), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3627).



**6.425.3.2** `template<typename E> virtual decaf::util::Iterator<E>*`  
`decaf::lang::Iterable< E >::iterator ()` [pure virtual]

**Returns:**

an iterator over a set of elements of type T.

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3721), `decaf::util::PriorityQueue< E >` (p. 3032), `decaf::util::StlList< E >` (p. 3598), `decaf::util::StlSet< E >` (p. 3628), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), `decaf::util::StlList< cms::Connection * >` (p. 3598), `decaf::util::StlSet< transport::TransportListener * >` (p. 3628), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3628), `decaf::util::StlSet< Resource * >` (p. 3628), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3628).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::clear()`, `decaf::util::AbstractCollection< cms::Connection * >::contains()`, `decaf::util::AbstractCollection< cms::Connection * >::copy()`, `decaf::util::AbstractCollection< cms::Connection * >::operator=()`, `decaf::util::AbstractCollection< cms::Connection * >::remove()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, `decaf::util::AbstractCollection< cms::Connection * >::removeAll()`, `decaf::util::AbstractCollection< cms::Connection * >::retainAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::toArray()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Iterable.h`

## 6.426 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

`#include <src/main/decaf/util/Iterator.h>` Inheritance diagram for `decaf::util::Iterator< T >`:

### Public Member Functions

- virtual `~Iterator ()`
- virtual `T next ()=0 throw ( lang::exceptions::NoSuchElementException )`  
*Returns the next element in the iteration.*
- virtual `bool hasNext () const =0`  
*Returns true if the iteration has more elements.*
- virtual `void remove ()=0 throw ( lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException )`  
*Removes from the underlying collection the last element returned by the iterator (optional operation).*

### 6.426.1 Detailed Description

`template<typename T> class decaf::util::Iterator< T >`

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

### 6.426.2 Constructor & Destructor Documentation

**6.426.2.1** `template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ()`  
`[inline, virtual]`

### 6.426.3 Member Function Documentation

**6.426.3.1** `template<typename T> virtual bool decaf::util::Iterator< T >::hasNext () const` `[pure virtual]`

Returns true if the iteration has more elements. Returns false if the next call to `next` would result in an `NoSuchElementException` to be thrown.

**6.426.3.2** `template<typename T> virtual T decaf::util::Iterator< T >::next ()`  
`throw ( lang::exceptions::NoSuchElementException )` `[pure virtual]`

Returns the next element in the iteration. Calling this method repeatedly until the `hasNext()` (p. 2154) method returns false will return each element in the underlying collection exactly once.

**Returns:**

next element in the iteration of elements

**Exceptions:**

*NoSuchElementException* - iteration has no more elements.

**6.426.3.3** `template<typename T> virtual void decaf::util::Iterator< T  
>::remove () throw ( lang::exceptions::IllegalStateException,  
lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation). This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

**Exceptions:**

*UnsupportedOperationException* - if the remove operation is not supported by this **Iterator** (p. 2154).

*IllegalStateException* - if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Iterator.h**

## 6.427 activemq::commands::JournalQueueAck Class Reference

#include <src/main/activemq/commands/JournalQueueAck.h> Inheritance diagram for activemq::commands::JournalQueueAck:

### Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalQueueAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &messageAck)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALQUEUEACK** = 52

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

## 6.427.1 Constructor & Destructor Documentation

**6.427.1.1** `activemq::commands::JournalQueueAck::JournalQueueAck ()`

**6.427.1.2** `virtual activemq::commands::JournalQueueAck::~~JournalQueueAck ()`  
[virtual]

## 6.427.2 Member Function Documentation

**6.427.2.1** `virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.427.2.2** `virtual void activemq::commands::JournalQueueAck::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.427.2.3** `virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

**6.427.2.4** `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.427.2.5 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ()`  
[virtual]
- 6.427.2.6 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () const`  
[virtual]
- 6.427.2.7 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ()`  
[virtual]
- 6.427.2.8 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () const` [virtual]
- 6.427.2.9 `virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.427.2.10 `virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & messageAck)` [virtual]
- 6.427.2.11 `virtual std::string activemq::commands::JournalQueueAck::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.833).

### 6.427.3 Field Documentation

- 6.427.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination`  
[protected]
- 6.427.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID _ - JOURNALQUEUEACK = 52` [static]
- 6.427.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

## 6.428 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2159).

#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.428.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2159).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.428.2 Constructor & Destructor Documentation

**6.428.2.1** `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.428.2.2** `virtual activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.428.3 Member Function Documentation

**6.428.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.428.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.428.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.428.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.428.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.428.3.6** virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.428.3.7** virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalQueueAckMarshaller.h**

## 6.429 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2163).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.429.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2163).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.429.2 Constructor & Destructor Documentation

**6.429.2.1** `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.429.2.2** `virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.429.3 Member Function Documentation

**6.429.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.429.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.429.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.429.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.429.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.429.3.6** virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.429.3.7** virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h

## 6.430 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2167).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.430.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2167).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.430.2 Constructor & Destructor Documentation

**6.430.2.1** `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.430.2.2** `virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.430.3 Member Function Documentation

**6.430.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.430.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.430.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.430.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.430.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.430.3.6** virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.430.3.7** virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h

## 6.431 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2171).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.431.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2171).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.431.2 Constructor & Destructor Documentation

**6.431.2.1** `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.431.2.2** `virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.431.3 Member Function Documentation

**6.431.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.431.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.431.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.431.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.431.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.431.3.6** virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.431.3.7** virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h

## 6.432 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2175).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.432.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2175).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.432.2 Constructor & Destructor Documentation

**6.432.2.1** `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.432.2.2** `virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.432.3 Member Function Documentation

**6.432.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.432.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.432.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.432.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.432.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.432.3.6** virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.432.3.7** virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalQueueAckMarshaller.h**

## 6.433 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2179).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.433.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2179).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.433.2 Constructor & Destructor Documentation

**6.433.2.1** `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

**6.433.2.2** `virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

## 6.433.3 Member Function Documentation

**6.433.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.433.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.433.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.433.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.433.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.433.3.6** virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.433.3.7** virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**

## 6.434 activemq::commands::JournalTopicAck Class Reference

#include <src/main/activemq/commands/JournalTopicAck.h> Inheritance diagram for activemq::commands::JournalTopicAck:

### Public Member Functions

- **JournalTopicAck** ()
- virtual **~JournalTopicAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTopicAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src) const  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long messageSequenceId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTOPICACK** = 50

## Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageId** > **messageId**
- long long **messageSequenceId**
- std::string **subscriptionName**
- std::string **clientId**
- **Pointer**< **TransactionId** > **transactionId**

## 6.434.1 Constructor & Destructor Documentation

**6.434.1.1** **activemq::commands::JournalTopicAck::JournalTopicAck ()**

**6.434.1.2** **virtual activemq::commands::JournalTopicAck::~~JournalTopicAck ()**  
[virtual]

## 6.434.2 Member Function Documentation

**6.434.2.1** **virtual JournalTopicAck\* activemq::commands::JournalTopicAck::cloneDataStructure () const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.434.2.2** **virtual void activemq::commands::JournalTopicAck::copyDataStructure (const DataStructure \* *src*)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.434.2.3** **virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure \* *value*) const** [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.



**6.434.2.4** `virtual std::string& activemq::commands::JournalTopicAck::getClientId () [virtual]`

**6.434.2.5** `virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const [virtual]`

**6.434.2.6** `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSetructure** (p. 1660) type copy.

Implements **activemq::commands::DataSetructure** (p. 1662).

- 
- 6.434.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ()`  
[virtual]
  - 6.434.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const`  
[virtual]
  - 6.434.2.9 `virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ()`  
[virtual]
  - 6.434.2.10 `virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () const` [virtual]
  - 6.434.2.11 `virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId () const`  
[virtual]
  - 6.434.2.12 `virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName ()`  
[virtual]
  - 6.434.2.13 `virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName () const`  
[virtual]
  - 6.434.2.14 `virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ()`  
[virtual]
  - 6.434.2.15 `virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () const`  
[virtual]
  - 6.434.2.16 `virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & clientId)` [virtual]
  - 6.434.2.17 `virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
  - 6.434.2.18 `virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
  - 6.434.2.19 `virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long messageSequenceId)` [virtual]
  - 6.434.2.20 `virtual void activemq::commands::JournalTopicAck::setSubscriptionName (const std::string & subscriptionName)` [virtual]
  - 6.434.2.21 `virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]
  - 6.434.2.22 `virtual std::string activemq::commands::JournalTopicAck::toString () const` [virtual]
- 

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 833).

**6.434.3 Field Documentation**

- 6.434.3.1** `std::string activemq::commands::JournalTopicAck::clientId` [protected]
- 6.434.3.2** `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination` [protected]
- 6.434.3.3** `const unsigned char activemq::commands::JournalTopicAck::ID_ - JOURNALTOPICACK = 50` [static]
- 6.434.3.4** `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId` [protected]
- 6.434.3.5** `long long activemq::commands::JournalTopicAck::messageSequenceId` [protected]
- 6.434.3.6** `std::string activemq::commands::JournalTopicAck::subscriptionName` [protected]
- 6.434.3.7** `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

## 6.435 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2188).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.435.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2188).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.435.2 Constructor & Destructor Documentation

**6.435.2.1** `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

**6.435.2.2** `virtual activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

## 6.435.3 Member Function Documentation

**6.435.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.435.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.435.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.435.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.435.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.435.3.6** virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.435.3.7** virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**

## 6.436 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2192).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.436.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2192).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.436.2 Constructor & Destructor Documentation

**6.436.2.1** `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` `[inline]`

**6.436.2.2** `virtual activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` `[inline, virtual]`

## 6.436.3 Member Function Documentation

**6.436.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.436.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.436.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.436.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.436.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.436.3.6** virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.436.3.7** virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**

## 6.437 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2196).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.437.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2196).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.437.2 Constructor & Destructor Documentation

**6.437.2.1** `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

**6.437.2.2** `virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

## 6.437.3 Member Function Documentation

**6.437.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.437.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.437.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.437.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.437.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.437.3.6** virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.437.3.7** virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTopicAckMarshaller.h**

## 6.438 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2200).

#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.438.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2200).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.438.2 Constructor & Destructor Documentation

**6.438.2.1** `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

**6.438.2.2** `virtual activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

## 6.438.3 Member Function Documentation

**6.438.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.438.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.438.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.438.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.438.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.438.3.6** virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.438.3.7** virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTopicAckMarshaller.h**

## 6.439 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2204).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.439.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2204).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.439.2 Constructor & Destructor Documentation

**6.439.2.1** `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` `[inline]`

**6.439.2.2** `virtual activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` `[inline, virtual]`

## 6.439.3 Member Function Documentation

**6.439.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.439.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.439.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.439.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.439.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.439.3.6** virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.439.3.7** virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTopicAckMarshaller.h**

## 6.440 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2208).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.440.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2208).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.440.2 Constructor & Destructor Documentation

**6.440.2.1** `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` `[inline]`

**6.440.2.2** `virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` `[inline, virtual]`

## 6.440.3 Member Function Documentation

**6.440.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.440.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.440.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.440.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.440.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.440.3.6** virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.440.3.7** virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**

## 6.441 activemq::commands::JournalTrace Class Reference

#include <src/main/activemq/commands/JournalTrace.h> Inheritance diagram for activemq::commands::JournalTrace:

### Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTrace \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTRACE** = 53

### Protected Attributes

- std::string **message**

## 6.441.1 Constructor & Destructor Documentation

**6.441.1.1** `activemq::commands::JournalTrace::JournalTrace ()`

**6.441.1.2** `virtual activemq::commands::JournalTrace::~~JournalTrace () [virtual]`

## 6.441.2 Member Function Documentation

**6.441.2.1** `virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.441.2.2** `virtual void activemq::commands::JournalTrace::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.441.2.3** `virtual bool activemq::commands::JournalTrace::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

**6.441.2.4** `virtual unsigned char activemq::commands::JournalTrace::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.441.2.5 `virtual std::string& activemq::commands::JournalTrace::getMessage ()`  
[virtual]
- 6.441.2.6 `virtual const std::string& activemq::commands::JournalTrace::getMessage`  
`() const` [virtual]
- 6.441.2.7 `virtual void activemq::commands::JournalTrace::setMessage (const`  
`std::string & message)` [virtual]
- 6.441.2.8 `virtual std::string activemq::commands::JournalTrace::toString () const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.833).

### 6.441.3 Field Documentation

- 6.441.3.1 `const unsigned char activemq::commands::JournalTrace::ID_ -`  
`JOURNALTRACE = 53` [static]
- 6.441.3.2 `std::string activemq::commands::JournalTrace::message` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

## 6.442 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2215).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.442.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2215).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.442.2 Constructor & Destructor Documentation

**6.442.2.1** `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.442.2.2** `virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.442.3 Member Function Documentation

**6.442.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.442.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.442.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.442.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.442.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.442.3.6** virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.442.3.7** virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**

## 6.443 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2219).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.443.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2219).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.443.2 Constructor & Destructor Documentation

**6.443.2.1** `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.443.2.2** `virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.443.3 Member Function Documentation

**6.443.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.443.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.443.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.443.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.443.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.443.3.6** virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.443.3.7** virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTraceMarshaller.h**

## 6.444 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2223).

#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.444.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2223).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.444.2 Constructor & Destructor Documentation

**6.444.2.1** `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.444.2.2** `virtual activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.444.3 Member Function Documentation

**6.444.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.444.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.444.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.444.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.444.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.444.3.6** virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.444.3.7** virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTraceMarshaller.h**

## 6.445 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2227).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.445.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2227).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.445.2 Constructor & Destructor Documentation

**6.445.2.1** `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.445.2.2** `virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.445.3 Member Function Documentation

**6.445.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.445.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.445.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.445.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.445.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.445.3.6** virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.445.3.7** virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTraceMarshaller.h**

## 6.446 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2231).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.446.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2231).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.446.2 Constructor & Destructor Documentation

**6.446.2.1** `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.446.2.2** `virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.446.3 Member Function Documentation

**6.446.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.446.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.446.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.446.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.446.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.446.3.6** virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.446.3.7** virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**

## 6.447 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2235).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.447.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **JournalTraceMarshaller** (p.2235).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.447.2 Constructor & Destructor Documentation

**6.447.2.1** `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

**6.447.2.2** `virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

## 6.447.3 Member Function Documentation

**6.447.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.447.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.447.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.447.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.447.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.447.3.6** virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.447.3.7** virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTraceMarshaller.h**

## 6.448 activemq::commands::JournalTransaction Class Reference

#include <src/main/activemq/commands/JournalTransaction.h> Inheritance diagram for activemq::commands::JournalTransaction:

### Public Member Functions

- **JournalTransaction** ()
- virtual **~JournalTransaction** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTransaction \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTRANSACTION** = 54

### Protected Attributes

- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**
- bool **wasPrepared**

## 6.448.1 Constructor & Destructor Documentation

6.448.1.1 `activemq::commands::JournalTransaction::JournalTransaction ()`

6.448.1.2 `virtual activemq::commands::JournalTransaction::~~JournalTransaction ()`  
[virtual]

## 6.448.2 Member Function Documentation

6.448.2.1 `virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure () const`  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

6.448.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

6.448.2.3 `virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

6.448.2.4 `virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).



- 6.448.2.5 virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () [virtual]
- 6.448.2.6 virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const [virtual]
- 6.448.2.7 virtual unsigned char activemq::commands::JournalTransaction::getType () const [virtual]
- 6.448.2.8 virtual bool activemq::commands::JournalTransaction::getWasPrepared () const [virtual]
- 6.448.2.9 virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]
- 6.448.2.10 virtual void activemq::commands::JournalTransaction::setType (unsigned char *type*) [virtual]
- 6.448.2.11 virtual void activemq::commands::JournalTransaction::setWasPrepared (bool *wasPrepared*) [virtual]
- 6.448.2.12 virtual std::string activemq::commands::JournalTransaction::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p.833).

### 6.448.3 Field Documentation

- 6.448.3.1 const unsigned char activemq::commands::JournalTransaction::ID\_ - JOURNALTRANSACTION = 54 [static]
- 6.448.3.2 Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId [protected]
- 6.448.3.3 unsigned char activemq::commands::JournalTransaction::type [protected]
- 6.448.3.4 bool activemq::commands::JournalTransaction::wasPrepared [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTransaction.h**

## 6.449 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2242).

#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.449.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2242).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.449.2 Constructor & Destructor Documentation

6.449.2.1 `activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::JournalTrans  
( ) [inline]`

6.449.2.2 `virtual  
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::~~JournalTra  
( ) [inline, virtual]`

## 6.449.3 Member Function Documentation

6.449.3.1 `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

6.449.3.2 `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

6.449.3.3 `virtual void ac-  
tivemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.449.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.449.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.449.3.6** virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.449.3.7** virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTransactionMarshaller.h**

## 6.450 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2246).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.450.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2246).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.450.2 Constructor & Destructor Documentation

**6.450.2.1** `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTrans  
 () [inline]`

**6.450.2.2** `virtual  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTra  
 () [inline, virtual]`

## 6.450.3 Member Function Documentation

**6.450.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject  
 () const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.450.3.2** `virtual unsigned char ac-  
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructu  
 () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.450.3.3** `virtual void ac-  
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal  
 (OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
 decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.450.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.450.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.450.3.6** virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.450.3.7** virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**

## 6.451 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2250).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.451.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2250).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.451.2 Constructor & Destructor Documentation

**6.451.2.1** `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTrans  
( ) [inline]`

**6.451.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTra  
( ) [inline, virtual]`

## 6.451.3 Member Function Documentation

**6.451.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.451.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.451.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.451.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.451.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.451.3.6** virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.451.3.7** virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**

## 6.452 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2254).

`#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller`:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.452.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2254).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.452.2 Constructor & Destructor Documentation

**6.452.2.1** `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTrans  
( ) [inline]`

**6.452.2.2** `virtual  
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTra  
( ) [inline, virtual]`

## 6.452.3 Member Function Documentation

**6.452.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.452.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructu  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.452.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.452.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.452.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.452.3.6** virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.452.3.7** virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTransactionMarshaller.h**

## 6.453 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2258).

`#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h>` Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller`:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.453.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2258).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.453.2 Constructor & Destructor Documentation

**6.453.2.1** `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTrans  
 () [inline]`

**6.453.2.2** `virtual  
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTra  
 () [inline, virtual]`

## 6.453.3 Member Function Documentation

**6.453.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject  
 () const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.453.3.2** `virtual unsigned char ac-  
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructu  
 () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.453.3.3** `virtual void ac-  
 tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal  
 (OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
 decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.453.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.453.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.453.3.6** virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.453.3.7** virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**

## 6.454 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2262).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.454.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2262).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.454.2 Constructor & Destructor Documentation

**6.454.2.1** `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTrans  
 () [inline]`

**6.454.2.2** `virtual  
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTra  
 () [inline, virtual]`

## 6.454.3 Member Function Documentation

**6.454.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject  
 () const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.454.3.2** `virtual unsigned char ac-  
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructu  
 () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.454.3.3** `virtual void ac-  
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal  
 (OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
 decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.454.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.454.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).



**6.454.3.6** virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.454.3.7** virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**

## 6.455 activemq::commands::KeepAliveInfo Class Reference

#include <src/main/activemq/commands/KeepAliveInfo.h> Inheritance diagram for activemq::commands::KeepAliveInfo:

### Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **KeepAliveInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_KEEPAALIVEINFO** = 10

## 6.455.1 Constructor & Destructor Documentation

**6.455.1.1** `activemq::commands::KeepAliveInfo::KeepAliveInfo ()`

**6.455.1.2** `virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo ()`  
[virtual]

## 6.455.2 Member Function Documentation

**6.455.2.1** `virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.455.2.2** `virtual void activemq::commands::KeepAliveInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.455.2.3** `virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.455.2.4** `virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

**6.455.2.5** `virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo () const [inline, virtual]`

**Returns:**

an answer of true to the **isKeepAliveInfo()** (p. 2268) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 761).

**6.455.2.6** `virtual std::string activemq::commands::KeepAliveInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.455.2.7** `virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

## 6.455.3 Field Documentation

**6.455.3.1** `const unsigned char activemq::commands::KeepAliveInfo::ID_ - KEEPALIVEINFO = 10 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/KeepAliveInfo.h`

## 6.456 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2269).

#include <src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.456.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2269).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.456.2 Constructor & Destructor Documentation

**6.456.2.1** `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.456.2.2** `virtual  
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.456.3 Member Function Documentation

**6.456.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.456.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.456.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 794).

**6.456.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 795).

**6.456.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 796).

**6.456.3.6** virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.456.3.7** virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h



## 6.457 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2273).

#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.457.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2273).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.457.2 Constructor & Destructor Documentation

**6.457.2.1** `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.457.2.2** `virtual  
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.457.3 Member Function Documentation

**6.457.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.457.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.457.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.457.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.457.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.457.3.6** virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.457.3.7** virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h

## 6.458 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2277).

#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.458.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2277).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.458.2 Constructor & Destructor Documentation

**6.458.2.1** `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.458.2.2** `virtual  
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.458.3 Member Function Documentation

**6.458.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.458.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.458.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.458.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.458.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.458.3.6** virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.458.3.7** virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h



## 6.459 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2281).

#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.459.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2281).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.459.2 Constructor & Destructor Documentation

**6.459.2.1** `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.459.2.2** `virtual  
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.459.3 Member Function Documentation

**6.459.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.459.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.459.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.459.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.459.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.459.3.6** virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.459.3.7** virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h

## 6.460 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2285).

#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.460.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2285).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.460.2 Constructor & Destructor Documentation

**6.460.2.1** `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.460.2.2** `virtual  
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.460.3 Member Function Documentation

**6.460.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.460.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.460.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.460.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.460.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.460.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.460.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h`



## 6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2289).

#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.461.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p.2289).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.461.2 Constructor & Destructor Documentation

**6.461.2.1** `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMar  
( ) [inline]`

**6.461.2.2** `virtual  
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoM  
( ) [inline, virtual]`

## 6.461.3 Member Function Documentation

**6.461.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.461.3.2** `virtual unsigned char ac-  
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureTy  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.461.3.3** `virtual void ac-  
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal  
(OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.461.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.461.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.461.3.6** virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.461.3.7** virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

## 6.462 decaf::security::Key Class Reference

The **Key** (p. 2293) interface is the top-level interface for all keys.

#include <src/main/decaf/security/Key.h> Inheritance diagram for decaf::security::Key:

### Public Member Functions

- virtual **~Key** ()
- virtual std::string **getAlgorithm** () const =0  
*Returns the standard algorithm name for this key.*
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0  
*Provides the key in its primary encoding format, or nothing if this key does not support encoding.*
- virtual std::string **getFormat** () const =0  
*Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.*

### 6.462.1 Detailed Description

The **Key** (p. 2293) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the getAlgorithm method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the getEncoded method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
algorithm AlgorithmIdentifier,
subjectPublicKey BIT STRING }
AlgorithmIdentifier ::= SEQUENCE {
algorithm OBJECT IDENTIFIER,
parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 2293) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

## 6.462.2 Constructor & Destructor Documentation

**6.462.2.1** `virtual decaf::security::Key::~~Key () [inline, virtual]`

## 6.462.3 Member Function Documentation

**6.462.3.1** `virtual std::string decaf::security::Key::getAlgorithm () const [pure virtual]`

Returns the standard algorithm name for this key. For example, "DSA" would indicate that this key is a DSA key.

### Returns:

the name of the algorithm associated with this key.

**6.462.3.2** `virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

### Parameters:

*output* Receives the encoded key, or nothing if the key does not support encoding.

**6.462.3.3** `virtual std::string decaf::security::Key::getFormat () const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding. The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is `SubjectPublicKeyInfo`, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is `PrivateKeyInfo`, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

### Returns:

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Key.h`

## 6.463 decaf::security::KeyException Class Reference

#include <src/main/decaf/security/KeyException.h> Inheritance diagram for decaf::security::KeyException:

### Public Member Functions

- **KeyException** () throw ()  
*Default Constructor.*
- **KeyException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **KeyException** (const **KeyException** &ex) throw ()  
*Copy Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **KeyException** (const std::exception \*cause) throw ()  
*Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **KeyException** \* clone () const  
*Clones this exception.*
- virtual ~**KeyException** () throw ()

### 6.463.1 Constructor & Destructor Documentation

#### 6.463.1.1 decaf::security::KeyException::KeyException () throw () [inline]

Default Constructor.

#### 6.463.1.2 decaf::security::KeyException::KeyException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

#### 6.463.1.3 decaf::security::KeyException::KeyException (const KeyException & *ex*) throw () [inline]

Copy Constructor.

##### Parameters:

*ex* An exception that should become this type of Exception

#### 6.463.1.4 decaf::security::KeyException::KeyException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

#### 6.463.1.5 decaf::security::KeyException::KeyException (const std::exception \* *cause*) throw () [inline]

Constructor.

##### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

#### 6.463.1.6 decaf::security::KeyException::KeyException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message



**6.463.1.7** `virtual decaf::security::KeyException::~~KeyException () throw ()`  
[inline, virtual]

## 6.463.2 Member Function Documentation

**6.463.2.1** `virtual KeyException* decaf::security::KeyException::clone () const`  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1973).

Reimplemented in `decaf::security::InvalidKeyException` (p. 2134), and `decaf::security::KeyManagementException` (p. 2300).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyException.h`

## 6.464 decaf::security::KeyManagementException Class Reference

#include <src/main/decaf/security/KeyManagementException.h> Inheritance diagram for decaf::security::KeyManagementException:

### Public Member Functions

- **KeyManagementException** () throw ()  
*Default Constructor.*
- **KeyManagementException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **KeyManagementException** (const **KeyManagementException** &ex) throw ()  
*Copy Constructor.*
- **KeyManagementException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **KeyManagementException** (const std::exception \*cause) throw ()  
*Constructor.*
- **KeyManagementException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **KeyManagementException** \* clone () const  
*Clones this exception.*
- virtual ~**KeyManagementException** () throw ()

### 6.464.1 Constructor & Destructor Documentation

#### 6.464.1.1 decaf::security::KeyManagementException::KeyManagementException () throw () [inline]

Default Constructor.

#### 6.464.1.2 decaf::security::KeyManagementException::KeyManagementException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.464.1.3 decaf::security::KeyManagementException::KeyManagementException (const KeyManagementException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.464.1.4 decaf::security::KeyManagementException::KeyManagementException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.464.1.5 decaf::security::KeyManagementException::KeyManagementException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.464.1.6 decaf::security::KeyManagementException::KeyManagementException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.464.1.7**    **virtual**  
**decaf::security::KeyManagementException::~~KeyManagementException**  
**() throw ()**    [inline, virtual]

## **6.464.2    Member Function Documentation**

**6.464.2.1**    **virtual KeyManagementException\* de-**  
**caf::security::KeyManagementException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2297).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyManagementException.h`

## 6.465 activemq::commands::LastPartialCommand Class Reference

#include <src/main/activemq/commands/LastPartialCommand.h> Inheritance diagram for activemq::commands::LastPartialCommand:

### Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **LastPartialCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*

### Static Public Attributes

- static const unsigned char **ID\_LASTPARTIALCOMMAND** = 61

#### 6.465.1 Constructor & Destructor Documentation

6.465.1.1 **activemq::commands::LastPartialCommand::LastPartialCommand** ()

6.465.1.2 **virtual**  
**activemq::commands::LastPartialCommand::~~LastPartialCommand** ()  
[virtual]

#### 6.465.2 Member Function Documentation

6.465.2.1 **virtual LastPartialCommand\* ac-**  
**tivemq::commands::LastPartialCommand::cloneDataStructure** () const  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Reimplemented from `activemq::commands::PartialCommand` (p. 2919).

**6.465.2.2** `virtual void activemq::commands::LastPartialCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

Reimplemented from `activemq::commands::PartialCommand` (p. 2919).

**6.465.2.3** `virtual bool activemq::commands::LastPartialCommand::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::PartialCommand` (p. 2919).

**6.465.2.4** `virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new `DataStructure` (p. 1660) type copy.

Reimplemented from `activemq::commands::PartialCommand` (p. 2920).

**6.465.2.5** `virtual std::string activemq::commands::LastPartialCommand::toString () const [virtual]`

Returns a string containing the information for this `DataStructure` (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::PartialCommand` (p. 2920).

### 6.465.3 Field Documentation

#### 6.465.3.1 `const unsigned char activemq::commands::LastPartialCommand::ID_ - LASTPARTIALCOMMAND = 61` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LastPartialCommand.h`

## 6.466 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2304).

#include <src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.466.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2304). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.466.2 Constructor & Destructor Documentation

**6.466.2.1** `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

**6.466.2.2** `virtual activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

## 6.466.3 Member Function Documentation

**6.466.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2923).

**6.466.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2923).

**6.466.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2923).

**6.466.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2924).

**6.466.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2924).

**6.466.3.6** virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2925).

**6.466.3.7** virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2925).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LastPartialCommandMarshaller.h**

## 6.467 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2308).

#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.467.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2308). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.467.2 Constructor & Destructor Documentation

**6.467.2.1** `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

**6.467.2.2** `virtual activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

## 6.467.3 Member Function Documentation

**6.467.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2935).

**6.467.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2935).

**6.467.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2935).

**6.467.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2936).

**6.467.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2936).

**6.467.3.6** virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2937).

**6.467.3.7** virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2937).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LastPartialCommandMarshaller.h**

## 6.468 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2312).

#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.468.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2312). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.468.2 Constructor & Destructor Documentation

**6.468.2.1** `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` `[inline]`

**6.468.2.2** `virtual activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` `[inline, virtual]`

## 6.468.3 Member Function Documentation

**6.468.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2927).

**6.468.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaller.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2927).

**6.468.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2927).

**6.468.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2928).

**6.468.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2928).

**6.468.3.6** virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2929).

**6.468.3.7** virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2929).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**

## 6.469 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2316).

#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.469.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2316). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.469.2 Constructor & Destructor Documentation

**6.469.2.1** `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` `[inline]`

**6.469.2.2** `virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` `[inline, virtual]`

## 6.469.3 Member Function Documentation

**6.469.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2931).

**6.469.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2931).

**6.469.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2931).

**6.469.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2932).

**6.469.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2932).

**6.469.3.6** virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2933).

**6.469.3.7** virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2933).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**LastPartialCommandMarshaller.h**

## 6.470 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2320).

#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.470.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2320). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.470.2 Constructor & Destructor Documentation

**6.470.2.1** `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

**6.470.2.2** `virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

## 6.470.3 Member Function Documentation

**6.470.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2939).

**6.470.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2939).

**6.470.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2939).

**6.470.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2940).

**6.470.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2940).

**6.470.3.6** virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2941).

**6.470.3.7** virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2941).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**

## 6.471 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2324).

#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.471.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **LastPartialCommandMarshaller** (p.2324). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.471.2 Constructor & Destructor Documentation

**6.471.2.1** `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

**6.471.2.2** `virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

## 6.471.3 Member Function Documentation

**6.471.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2943).

**6.471.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2943).

**6.471.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2943).

**6.471.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2944).

**6.471.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2944).

**6.471.3.6** virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2945).

**6.471.3.7** virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2945).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**

## 6.472 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 2328) **Comparator** (p. 1217) that compares to elements to determine if the first is less than the second.

#include <src/main/decaf/util/comparators/Less.h> Inheritance diagram for decaf::util::comparators::Less< E >:

### Public Member Functions

- **Less** ()
- virtual **~Less** ()
- virtual bool **operator**() (const E &left, const E &right) const  
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1217) to be passed to an STL **Map** (p. 2459) for use as the sorting criteria.*
- virtual int **compare** (const E &o1, const E &o2) const  
*Compares its two arguments for order.*

### 6.472.1 Detailed Description

template<typename E> class decaf::util::comparators::Less< E >

Simple **Less** (p. 2328) **Comparator** (p. 1217) that compares to elements to determine if the first is less than the second. This can be used in **Collection** (p. 1184) classes to sort elements according to their natural ordering. By design the Comparator's compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

Since:

1.0

### 6.472.2 Constructor & Destructor Documentation

6.472.2.1 template<typename E > decaf::util::comparators::Less< E >::Less ()  
[inline]

6.472.2.2 template<typename E > virtual decaf::util::comparators::Less< E >::~~Less () [inline, virtual]

### 6.472.3 Member Function Documentation

6.472.3.1 template<typename E > virtual int decaf::util::comparators::Less< E >::compare (const E & o1, const E & o2) const [inline, virtual]

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.



The implementor must ensure that  $\text{sgn}(\text{compare}(x, y)) == -\text{sgn}(\text{compare}(y, x))$  for all  $x$  and  $y$ . (This implies that  $\text{compare}(x, y)$  must throw an exception if and only if  $\text{compare}(y, x)$  throws an exception.)

The implementor must also ensure that the relation is transitive:  $((\text{compare}(x, y) > 0) \ \&\& \ (\text{compare}(y, z) > 0))$  implies  $\text{compare}(x, z) > 0$ .

Finally, the implementer must ensure that  $\text{compare}(x, y) == 0$  implies that  $\text{sgn}(\text{compare}(x, z)) == \text{sgn}(\text{compare}(y, z))$  for all  $z$ .

It is generally the case, but not strictly required that  $(\text{compare}(x, y) == 0) == (x == y)$ . Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

#### Parameters:

- o1* - the first object to be compared
- o2* - the second object to be compared

#### Returns:

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements **decaf::util::Comparator< E >** (p.1217).

#### 6.472.3.2 `template<typename E> virtual bool decaf::util::comparators::Less< E >::operator() (const E & left, const E & right) const [inline, virtual]`

Implementation of the Binary function interface as a means of allowing a **Comparator** (p.1217) to be passed to an STL **Map** (p.2459) for use as the sorting criteria.

#### Parameters:

- left* - the Left hand side operand.
- right* - the Right hand side operand.

#### Returns:

true if the value of left is less than the value of right.

Implements **decaf::util::Comparator< E >** (p.1218).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/Less.h`

## 6.473 std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

#include <src/main/decaf/lang/ArrayPointer.h>Inheritance diagram for std::less< decaf::lang::ArrayPointer< T > >:

### Public Member Functions

- **bool operator()** (const **decaf::lang::ArrayPointer< T >** &left, const **decaf::lang::ArrayPointer< T >** &right) const

#### 6.473.1 Detailed Description

template<typename T> struct std::less< decaf::lang::ArrayPointer< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

#### 6.473.2 Member Function Documentation

- 6.473.2.1** template<typename T > bool std::less< decaf::lang::ArrayPointer< T > >::operator() (const decaf::lang::ArrayPointer< T > & *left*, const decaf::lang::ArrayPointer< T > & *right*) const [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

## 6.474 `std::less< decaf::lang::Pointer< T > >` Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

`#include <src/main/decaf/lang/Pointer.h>`Inheritance diagram for `std::less< decaf::lang::Pointer< T > >`:

### Public Member Functions

- `bool operator() (const decaf::lang::Pointer< T > &left, const decaf::lang::Pointer< T > &right) const`

#### 6.474.1 Detailed Description

`template<typename T> struct std::less< decaf::lang::Pointer< T > >`

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

#### 6.474.2 Member Function Documentation

- 6.474.2.1** `template<typename T > bool std::less< decaf::lang::Pointer< T > >::operator() (const decaf::lang::Pointer< T > & left, const decaf::lang::Pointer< T > & right) const` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.475 decaf::util::logging::Level Class Reference

The **Level** (p. 2332) class defines a set of standard **logging** (p. 175) levels that can be used to control **logging** (p. 175) output.

#include <src/main/decaf/util/logging/Level.h> Inheritance diagram for decaf::util::logging::Level:

### Public Member Functions

- virtual **~Level** ()
- int **intValue** () const
- std::string **getName** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Level** &value) const
- virtual bool **equals** (const **Level** &value) const
- virtual bool **operator==** (const **Level** &value) const
- virtual bool **operator<** (const **Level** &value) const

### Static Public Member Functions

- static **Level** **parse** (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse a level name string into a **Level** (p. 2332).*

### Static Public Attributes

- static const **Level** **INHERIT**  
*NULL is a special level that indicates that the **Logger** (p. 2386) should get its **Level** (p. 2332) from its parent **Logger** (p. 2386), the value is initialized as zero.*
- static const **Level** **OFF**  
*OFF is a special level that can be used to turn off **logging** (p. 175).*
- static const **Level** **SEVERE**  
*SEVERE is a message level indicating a serious failure.*
- static const **Level** **WARNING**  
*WARNING is a message level indicating a potential problem.*
- static const **Level** **INFO**  
*INFO is a message level for informational messages.*
- static const **Level** **DEBUG**  
*DEBUG is a level for more verbose informative messages.*
- static const **Level** **CONFIG**

*CONFIG is a message level for static configuration messages.*

- static const **Level FINE**

*FINE is a message level providing tracing information.*

- static const **Level FINER**

*FINER indicates a fairly detailed tracing message.*

- static const **Level FINEST**

*FINEST indicates a highly detailed tracing message.*

- static const **Level ALL**

*ALL indicates that all messages should be logged.*

## Protected Member Functions

- **Level** (const std::string &name, int value)

*Create a named **Level** (p. 2332) with a given integer value.*

### 6.475.1 Detailed Description

The **Level** (p. 2332) class defines a set of standard **logging** (p. 175) levels that can be used to control **logging** (p. 175) output. The **logging** (p. 175) **Level** (p. 2332) objects are ordered and are specified by ordered integers. Enabling **logging** (p. 175) at a given level also enables **logging** (p. 175) at all higher levels.

Clients should normally use the predefined **Level** (p. 2332) constants such as **Level.SEVERE** (p. 2336).

The levels in descending order are:

\* SEVERE (highest value) \* WARNING \* INFO \* DEBUG \* CONFIG \* FINE \* FINER \* FINEST (lowest value)

In addition there is a level OFF that can be used to turn off **logging** (p. 175), and a level ALL that can be used to enable **logging** (p. 175) of all messages.

It is possible for third parties to define additional **logging** (p. 175) levels by subclassing **Level** (p. 2332). In such cases subclasses should take care to chose unique integer level values.

**Since:**

1.0

### 6.475.2 Constructor & Destructor Documentation

#### 6.475.2.1 decaf::util::logging::Level::Level (const std::string & name, int value) [protected]

Create a named **Level** (p. 2332) with a given integer value.

**Parameters:**

- name* Name of the level, e.g. SEVERE  
*value* Unique integer value of this level, e.g. 100

**6.475.2.2** virtual decaf::util::logging::Level::~~Level () [virtual]

**6.475.3 Member Function Documentation**

**6.475.3.1** virtual int decaf::util::logging::Level::compareTo (const Level & *value*) const [virtual]

**6.475.3.2** virtual bool decaf::util::logging::Level::equals (const Level & *value*) const [virtual]

**6.475.3.3** std::string decaf::util::logging::Level::getName () const [inline]

**Returns:**

the name of this **Level** (p. 2332) instance.

**6.475.3.4** int decaf::util::logging::Level::intValue () const [inline]

**Returns:**

the integer value of this level instance.

**6.475.3.5** virtual bool decaf::util::logging::Level::operator< (const Level & *value*) const [virtual]

**6.475.3.6** virtual bool decaf::util::logging::Level::operator== (const Level & *value*) const [virtual]

**6.475.3.7** static Level decaf::util::logging::Level::parse (const std::string & *name*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Parse a level name string into a **Level** (p. 2332). The argument string may consist of either a level name or an integer value.

For example:

```
* "SEVERE" * "1000"
```

**Parameters:**

*name* - The name or int value of the desired **Level** (p. 2332)

**Returns:**

the parsed **Level** (p. 2332) value, passing in a level name that is an int value that is not one of the known **Level** (p. 2332) values will result in a new **Level** (p. 2332) that has been initialized with that int value and name as the string form of the int.

**Exceptions:**

***IllegalArgumentException*** if the value is not valid, validity means that the string is either a valid int (between Integer::MIN\_VALUE and Integer::MAX\_VALUE or is one of the known level names.

**6.475.3.8 std::string decaf::util::logging::Level::toString () const [inline]****Returns:**

the string value of this **Level** (p. 2332), e.g. "SEVERE".

**6.475.4 Field Documentation****6.475.4.1 const Level decaf::util::logging::Level::ALL [static]**

ALL indicates that all messages should be logged. This level is initialized to Integer::MIN\_VALUE.

**6.475.4.2 const Level decaf::util::logging::Level::CONFIG [static]**

CONFIG is a message level for static configuration messages. CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

**6.475.4.3 const Level decaf::util::logging::Level::DEBUG [static]**

DEBUG is a level for more verbose informative messages. DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

**6.475.4.4 const Level decaf::util::logging::Level::FINE [static]**

FINE is a message level providing tracing information. All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth **logging** (p. 175) as FINE. This level is initialized to 500.

**6.475.4.5 const Level decaf::util::logging::Level::FINER [static]**

FINER indicates a fairly detailed tracing message. By default **logging** (p. 175) calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

**6.475.4.6    const Level decaf::util::logging::Level::FINEST    [static]**

FINEST indicates a highly detailed tracing message. This level is initialized to 300.

**6.475.4.7    const Level decaf::util::logging::Level::INFO    [static]**

INFO is a message level for informational messages. Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

**6.475.4.8    const Level decaf::util::logging::Level::INHERIT    [static]**

NULL is a special level that indicates that the **Logger** (p. 2386) should get its **Level** (p. 2332) from its parent **Logger** (p. 2386), the value is initialized as zero.

**6.475.4.9    const Level decaf::util::logging::Level::OFF    [static]**

OFF is a special level that can be used to turn off **logging** (p. 175). This level is initialized to `Integer::MAX_VALUE`

**6.475.4.10    const Level decaf::util::logging::Level::SEVERE    [static]**

SEVERE is a message level indicating a serious failure. In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

**6.475.4.11    const Level decaf::util::logging::Level::WARNING    [static]**

WARNING is a message level indicating a potential problem. In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Level.h`



## 6.476 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

#include <src/main/decaf/util/List.h> Inheritance diagram for decaf::util::List< E >:

### Public Member Functions

- **List** ()
- virtual **~List** ()
- virtual **ListIterator**< E > \* **listIterator** ()=0
- virtual **ListIterator**< E > \* **listIterator** () const =0
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **indexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual std::size\_t **lastIndexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual E **get** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Gets the element contained at position passed.*
- virtual E **set** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Replaces the element at the specified position in this list with the specified element.*
- virtual void **add** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Inserts the specified element at the specified position in this list.*
- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Inserts all of the elements in the specified collection into this list at the specified position (optional operation).*
- virtual E **remove** (std::size\_t index)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Removes the element at the specified position in this list.*

## 6.476.1 Detailed Description

**template<typename E> class decaf::util::List< E >**

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements *e1* and *e2* such that *e1.equals(e2)*, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

## 6.476.2 Constructor & Destructor Documentation

**6.476.2.1** **template<typename E> decaf::util::List< E >::List ()** [inline]

**6.476.2.2** **template<typename E> virtual decaf::util::List< E >::~~List ()** [inline, virtual]

## 6.476.3 Member Function Documentation

**6.476.3.1** **template<typename E> virtual void decaf::util::List< E >::add (std::size\_t *index*, const E & *element*)** throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

### Parameters:

*index* - index at which the specified element is to be inserted

*element* - element to be inserted

### Exceptions:

*IndexOutOfBoundsException* - if the index is greater than size

*UnsupportedOperationException* - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 3594), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3594), **decaf::util::StlList< CompositeTask \* >** (p. 3594), **decaf::util::StlList< URI >** (p. 3594), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3594), **decaf::util::StlList< PrimitiveValueNode >** (p. 3594), **decaf::util::StlList< Pointer< Command > >** (p. 3594), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3594), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3594), **decaf::util::StlList< cms::Destination \* >** (p. 3594), **decaf::util::StlList< cms::Session \* >** (p. 3594), and **decaf::util::StlList< cms::Connection \* >** (p. 3594).

**6.476.3.2** `template<typename E> virtual bool decaf::util::List< E >::addAll (std::size_t index, const Collection< E > & source) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation). Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

**Parameters:**

*index* The index at which to insert the first element from the specified collection  
*source* The **Collection** (p. 1184) containing elements to be added to this list

**Returns:**

true if this list changed as a result of the call

**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size  
*UnsupportedOperationException* - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3595), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3595), `decaf::util::StlList< CompositeTask * >` (p. 3595), `decaf::util::StlList< URI >` (p. 3595), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3595), `decaf::util::StlList< PrimitiveValueNode >` (p. 3595), `decaf::util::StlList< Pointer< Command > >` (p. 3595), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3595), `decaf::util::StlList< cms::MessageProducer * >` (p. 3595), `decaf::util::StlList< cms::Destination * >` (p. 3595), `decaf::util::StlList< cms::Session * >` (p. 3595), and `decaf::util::StlList< cms::Connection * >` (p. 3595).

**6.476.3.3** `template<typename E> virtual E decaf::util::List< E >::get (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Gets the element contained at position passed.

**Parameters:**

*index* - position to get

**Returns:**

value at index

Implemented in `decaf::util::StlList< E >` (p. 3597), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3597), `decaf::util::StlList< CompositeTask * >` (p. 3597), `decaf::util::StlList< URI >` (p. 3597), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3597), `decaf::util::StlList< PrimitiveValueNode >` (p. 3597), `decaf::util::StlList< Pointer< Command > >` (p. 3597), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3597), `decaf::util::StlList< cms::MessageProducer * >` (p. 3597), `decaf::util::StlList< cms::Destination * >` (p. 3597), `decaf::util::StlList< cms::Session * >` (p. 3597), and `decaf::util::StlList< cms::Connection * >` (p. 3597).

**6.476.3.4** `template<typename E> virtual std::size_t decaf::util::List<E>::indexOf (const E & value) throw ( decaf::lang::exceptions::NoSuchElementException ) [pure virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

**Parameters:**

*value* - element to search for

**Returns:**

the index of the first occurrence of the specified element in this list,

**Exceptions:**

*NoSuchElementException* if *value* is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3597), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3597), `decaf::util::StlList< CompositeTask * >` (p. 3597), `decaf::util::StlList< URI >` (p. 3597), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3597), `decaf::util::StlList< PrimitiveValueNode >` (p. 3597), `decaf::util::StlList< Pointer< Command > >` (p. 3597), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3597), `decaf::util::StlList< cms::MessageProducer * >` (p. 3597), `decaf::util::StlList< cms::Destination * >` (p. 3597), `decaf::util::StlList< cms::Session * >` (p. 3597), and `decaf::util::StlList< cms::Connection * >` (p. 3597).

**6.476.3.5** `template<typename E> virtual size_t decaf::util::List<E>::lastIndexOf (const E & value) throw ( decaf::lang::exceptions::NoSuchElementException ) [pure virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

**Parameters:**

*value* - element to search for

**Returns:**

the index of the last occurrence of the specified element in this list.

**Exceptions:**

*NoSuchElementException* if *value* is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3598), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), and `decaf::util::StlList< cms::Connection * >` (p. 3598).

**6.476.3.6** `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 3598), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3598), `decaf::util::StlList< CompositeTask * >` (p. 3598), `decaf::util::StlList< URI >` (p. 3598), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3598), `decaf::util::StlList< PrimitiveValueNode >` (p. 3598), `decaf::util::StlList< Pointer< Command > >` (p. 3598), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3598), `decaf::util::StlList< cms::MessageProducer * >` (p. 3598), `decaf::util::StlList< cms::Destination * >` (p. 3598), `decaf::util::StlList< cms::Session * >` (p. 3598), and `decaf::util::StlList< cms::Connection * >` (p. 3598).

**6.476.3.7** `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

#### Parameters:

*index* index of first element to be returned from the list iterator (by a call to the next method).

#### Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

#### Exceptions:

*IndexOutOfBoundsException* if the index is out of range (`index < 0 || index > size()` (p. 1192))

Implemented in `decaf::util::StlList< E >` (p. 3599), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3599), `decaf::util::StlList< CompositeTask * >` (p. 3599), `decaf::util::StlList< URI >` (p. 3599), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3599), `decaf::util::StlList< PrimitiveValueNode >` (p. 3599), `decaf::util::StlList< Pointer< Command > >` (p. 3599), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3599), `decaf::util::StlList< cms::MessageProducer * >` (p. 3599), `decaf::util::StlList< cms::Destination * >` (p. 3599), `decaf::util::StlList< cms::Session * >` (p. 3599), and `decaf::util::StlList< cms::Connection * >` (p. 3599).

**6.476.3.8** `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () const [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 3599), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3599), `decaf::util::StlList< CompositeTask * >` (p. 3599), `decaf::util::StlList< URI >` (p. 3599), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3599), `decaf::util::StlList< PrimitiveValueNode >` (p. 3599), `decaf::util::StlList< Pointer< Command > >` (p. 3599), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3599), `decaf::util::StlList< cms::MessageProducer * >` (p. 3599), `decaf::util::StlList< cms::Destination * >` (p. 3599), `decaf::util::StlList< cms::Session * >` (p. 3599), and `decaf::util::StlList< cms::Connection * >` (p. 3599).

**6.476.3.9** `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () [pure virtual]`

**Returns:**

a list iterator over the elements in this list (in proper sequence).

Implemented in `decaf::util::StlList< E >` (p. 3599), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3599), `decaf::util::StlList< CompositeTask * >` (p. 3599), `decaf::util::StlList< URI >` (p. 3599), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3599), `decaf::util::StlList< PrimitiveValueNode >` (p. 3599), `decaf::util::StlList< Pointer< Command > >` (p. 3599), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3599), `decaf::util::StlList< cms::MessageProducer * >` (p. 3599), `decaf::util::StlList< cms::Destination * >` (p. 3599), `decaf::util::StlList< cms::Session * >` (p. 3599), and `decaf::util::StlList< cms::Connection * >` (p. 3599).

**6.476.3.10** `template<typename E> virtual E decaf::util::List< E >::remove (std::size_t index) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

**Parameters:**

*index* - the index of the element to be removed

**Returns:**

the element previously at the specified position

**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size

*UnsupportedOperationException* - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3599), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3599), `decaf::util::StlList< CompositeTask * >` (p. 3599), `decaf::util::StlList< URI >` (p. 3599), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3599), `decaf::util::StlList< PrimitiveValueNode >` (p. 3599), `decaf::util::StlList< Pointer< Command > >` (p. 3599), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3599), `decaf::util::StlList< cms::MessageProducer * >` (p. 3599), `decaf::util::StlList< cms::Destination * >` (p. 3599), `decaf::util::StlList< cms::Session * >` (p. 3599), and `decaf::util::StlList< cms::Connection * >` (p. 3599).

**6.476.3.11** `template<typename E> virtual E decaf::util::List< E >::set (std::size_t index, const E & element) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Replaces the element at the specified position in this list with the specified element.

**Parameters:**

*index* - index of the element to replace

*element* - element to be stored at the specified position

**Returns:**

the element previously at the specified position

**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size

Implemented in `decaf::util::StlList< E >` (p. 3600), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3600), `decaf::util::StlList< CompositeTask * >` (p. 3600), `decaf::util::StlList< URI >` (p. 3600), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3600), `decaf::util::StlList< PrimitiveValueNode >` (p. 3600), `decaf::util::StlList< Pointer< Command > >` (p. 3600), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3600), `decaf::util::StlList< cms::MessageProducer * >` (p. 3600), `decaf::util::StlList< cms::Destination * >` (p. 3600), `decaf::util::StlList< cms::Session * >` (p. 3600), and `decaf::util::StlList< cms::Connection * >` (p. 3600).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/List.h`

## 6.477 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h> Inheritance diagram for
decaf::util::ListIterator< E >:
```

### Public Member Functions

- virtual **~ListIterator** ()
- virtual void **add** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the list (optional operation).*
- virtual void **set** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Replaces the last element returned by next or previous with the specified element (optional operation).*
- virtual bool **hasPrevious** () const =0  
*Returns true if this list iterator has more elements when traversing the list in the reverse direction.*
- virtual E **previous** ()=0  
*Returns the previous element in the list.*
- virtual int **nextIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to next.*
- virtual int **previousIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to previous.*

### 6.477.1 Detailed Description

```
template<typename E> class decaf::util::ListIterator< E >
```

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 2155) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 2154) or **previous()** (p. 2346).



## 6.477.2 Constructor & Destructor Documentation

**6.477.2.1** `template<typename E > virtual decaf::util::ListIterator< E >::~~ListIterator () [inline, virtual]`

## 6.477.3 Member Function Documentation

**6.477.3.1** `template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & e) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Inserts the specified element into the list (optional operation). The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

### Parameters:

*e* - the element to insert.

### Exceptions:

*UnsupportedOperationException* - if the add method is not supported by this list iterator.

*IllegalArgumentException* - if some aspect of this element prevents it from being added to this list.

**6.477.3.2** `template<typename E > virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction. (In other words, returns true if previous would return an element rather than throwing an exception.)

### Returns:

true if the list iterator has more elements when traversing the list in the reverse direction.

**6.477.3.3** `template<typename E > virtual int decaf::util::ListIterator< E >::nextIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next. (Returns list size if the list iterator is at the end of the list.)

### Returns:

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

**6.477.3.4** `template<typename E > virtual E decaf::util::ListIterator< E >::previous  
( ) [pure virtual]`

Returns the previous element in the list. This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to `next` to go back and forth. (Note that alternating calls to `next` and `previous` will return the same element repeatedly.)

**Returns:**

the previous element in the list.

**Exceptions:**

*NoSuchElementException* - if the iteration has no previous element.

**6.477.3.5** `template<typename E > virtual int decaf::util::ListIterator< E  
>::previousIndex ( ) const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to `previous`. (Returns -1 if the list iterator is at the beginning of the list.)

**Returns:**

the index of the element that would be returned by a subsequent call to `previous`, or -1 if list iterator is at beginning of list.

**6.477.3.6** `template<typename E > virtual void  
decaf::util::ListIterator< E >::set (const E & e) throw (  
decaf::lang::exceptions::UnsupportedOperationException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::IllegalStateException ) [pure virtual]`

Replaces the last element returned by `next` or `previous` with the specified element (optional operation). This call can be made only if neither **ListIterator.remove** (p. 2155) nor **ListIterator.add** (p. 2345) have been called after the last call to `next` or `previous`.

**Parameters:**

*e* - the element with which to replace the last element returned by `next` or `previous`.

**Exceptions:**

*UnsupportedOperationException* - if the add method is not supported by this list iterator.

*IllegalArgumentException* - if some aspect of this element prevents it from being added to this list.

*IllegalStateException* - if neither `next` nor `previous` have been called, or `remove` or `add` have been called after the last call to `next` or `previous`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

## 6.478 activemq::commands::LocalTransactionId Class Reference

#include <src/main/activemq/commands/LocalTransactionId.h> Inheritance diagram for activemq::commands::LocalTransactionId:

### Public Types

- typedef decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR

### Public Member Functions

- LocalTransactionId ()
- LocalTransactionId (const LocalTransactionId &other)
- virtual ~LocalTransactionId ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual LocalTransactionId \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const DataStructure \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this DataStructure (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const DataStructure \*value) const  
*Compares the DataStructure (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual const Pointer< ConnectionId > & **getConnectionId** () const
- virtual Pointer< ConnectionId > & **getConnectionId** ()
- virtual void **setConnectionId** (const Pointer< ConnectionId > &connectionId)
- virtual int **compareTo** (const LocalTransactionId &value) const
- virtual bool **equals** (const LocalTransactionId &value) const
- virtual bool **operator==** (const LocalTransactionId &value) const
- virtual bool **operator<** (const LocalTransactionId &value) const
- LocalTransactionId & **operator=** (const LocalTransactionId &other)

### Static Public Attributes

- static const unsigned char **ID\_LOCALTRANSACTIONID** = 111

## Protected Attributes

- long long **value**
- **Pointer< ConnectionId > connectionId**

### 6.478.1 Member Typedef Documentation

**6.478.1.1** `typedef decaf::lang::PointerComparator<LocalTransactionId>  
activemq::commands::LocalTransactionId::COMPARATOR`

Reimplemented from `activemq::commands::TransactionId` (p. 3819).

### 6.478.2 Constructor & Destructor Documentation

**6.478.2.1** `activemq::commands::LocalTransactionId::LocalTransactionId ()`

**6.478.2.2** `activemq::commands::LocalTransactionId::LocalTransactionId (const  
LocalTransactionId & other)`

**6.478.2.3** `virtual activemq::commands::LocalTransactionId::~~LocalTransactionId ()  
[virtual]`

### 6.478.3 Member Function Documentation

**6.478.3.1** `virtual LocalTransactionId* ac-  
tivemq::commands::LocalTransactionId::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.478.3.2** `virtual int activemq::commands::LocalTransactionId::compareTo (const  
LocalTransactionId & value) const [virtual]`

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.478.3.3** `virtual void activemq::commands::LocalTransactionId::copyDataStructure  
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.478.3.4** `virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.478.3.5** `virtual bool activemq::commands::LocalTransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.478.3.6** `virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ()` [virtual]

**6.478.3.7** `virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const` [virtual]

**6.478.3.8** `virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.478.3.9** `virtual long long activemq::commands::LocalTransactionId::getValue () const` [virtual]

**6.478.3.10** `virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.478.3.11** `LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & other)`

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.478.3.12** `virtual bool activemq::commands::LocalTransactionId::operator==(const LocalTransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.478.3.13** `virtual void activemq::commands::LocalTransactionId::setConnectionId(const Pointer< ConnectionId > & connectionId)` [virtual]

**6.478.3.14** `virtual void activemq::commands::LocalTransactionId::setValue (long long value)` [virtual]

**6.478.3.15** `virtual std::string activemq::commands::LocalTransactionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

## 6.478.4 Field Documentation

**6.478.4.1** `Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId` [protected]

**6.478.4.2** `const unsigned char activemq::commands::LocalTransactionId::ID_ - LOCALTRANSACTIONID = 111` [static]

**6.478.4.3** `long long activemq::commands::LocalTransactionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LocalTransactionId.h`

## 6.479 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2351).

#include <src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.479.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2351).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.479.2 Constructor & Destructor Documentation

**6.479.2.1** `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.479.2.2** `virtual activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.479.3 Member Function Documentation

**6.479.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.479.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.479.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3844).

**6.479.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3844).

**6.479.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3845).

**6.479.3.6** virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3845).

**6.479.3.7** virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3846).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LocalTransactionIdMarshaller.h**

## 6.480 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2355).

#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.480.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2355).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.480.2 Constructor & Destructor Documentation

**6.480.2.1** `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.480.2.2** `virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.480.3 Member Function Documentation

**6.480.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.480.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.480.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3832).

**6.480.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3832).

**6.480.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3833).

**6.480.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3833).

**6.480.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3834).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h`

## 6.481 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2359).

#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.481.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2359).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.481.2 Constructor & Destructor Documentation

**6.481.2.1** `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.481.2.2** `virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.481.3 Member Function Documentation

**6.481.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.481.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.481.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3836).

**6.481.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3836).

**6.481.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3837).

**6.481.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3837).

**6.481.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3838).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h`

## 6.482 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2363).

#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.482.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2363).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.482.2 Constructor & Destructor Documentation

**6.482.2.1** `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.482.2.2** `virtual activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.482.3 Member Function Documentation

**6.482.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.482.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.482.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3824).

**6.482.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3824).

**6.482.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3825).

**6.482.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3825).

**6.482.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3826).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h`

## 6.483 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2367).

#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.483.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2367).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.483.2 Constructor & Destructor Documentation

**6.483.2.1** `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.483.2.2** `virtual activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.483.3 Member Function Documentation

**6.483.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.483.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.483.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3840).

**6.483.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3840).

**6.483.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3841).

**6.483.3.6** virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3841).

**6.483.3.7** virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3842).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LocalTransactionIdMarshaller.h**

## 6.484 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2371).

#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.484.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2371).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.484.2 Constructor & Destructor Documentation

**6.484.2.1** `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

**6.484.2.2** `virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

## 6.484.3 Member Function Documentation

**6.484.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.484.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.484.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3828).

**6.484.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3828).

**6.484.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3829).

**6.484.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3829).

**6.484.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3830).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h`

## 6.485 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

### Public Member Functions

- **Lock** (**Synchronizable** \*object, const bool initiallyLocked=true)  
*Constructor - initializes the object member and **locks** (p. 174) the object if desired.*
- virtual ~**Lock** ()  
*Destructor - Unlocks the object if it is locked.*
- void **lock** ()  
*Locks the object.*
- void **unlock** ()  
*Unlocks the object if it is already locked, otherwise a call to this method has no effect.*
- bool **isLocked** () const  
*Indicates whether or not the object is locked.*

### 6.485.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

**Since:**

1.0

### 6.485.2 Constructor & Destructor Documentation

#### 6.485.2.1 decaf::util::concurrent::Lock::Lock (**Synchronizable** \* object, const bool *initiallyLocked* = true)

Constructor - initializes the object member and **locks** (p. 174) the object if desired.

**Parameters:**

*object* The sync object to control

*initiallyLocked* If true, the object will automatically be locked.

#### 6.485.2.2 virtual decaf::util::concurrent::Lock::~~Lock () [virtual]

Destructor - Unlocks the object if it is locked.

### 6.485.3 Member Function Documentation

#### 6.485.3.1 `bool decaf::util::concurrent::Lock::isLocked () const [inline]`

Indicates whether or not the object is locked.

**Returns:**

true if the object is locked, otherwise false.

#### 6.485.3.2 `void decaf::util::concurrent::Lock::lock ()`

Locks the object.

#### 6.485.3.3 `void decaf::util::concurrent::Lock::unlock ()`

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Lock.h`



## 6.486 decaf::util::concurrent::locks::Lock Class Reference

**Lock** (p. 2377) implementations provide more extensive locking operations than can be obtained using synchronized statements.

#include <src/main/decaf/util/concurrent/locks/Lock.h> Inheritance diagram for decaf::util::concurrent::locks::Lock:

### Public Member Functions

- virtual **~Lock** ()
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is free at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Releases the lock.*
- virtual **Condition** \* **newCondition** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a new **Condition** (p. 1249) instance that is bound to this **Lock** (p. 2377) instance.*

### 6.486.1 Detailed Description

**Lock** (p. 2377) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1249) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some **locks** (p. 174) may allow **concurrent** (p. 171) access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 3172).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor **locks** (p. 174), and helps avoid many common programming errors involving **locks**

(p. 174), there are occasions where you need to work with **locks** (p. 174) in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 2377) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple **locks** (p. 174) to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of **locks** (p. 174) that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 2377) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all **code** (p. 1183) that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

**Lock** (p. 2377) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 2381)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 2379), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 2377) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 2377) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

## 6.486.2 Constructor & Destructor Documentation

**6.486.2.1** `virtual decaf::util::concurrent::locks::Lock::~~Lock () [inline, virtual]`

## 6.486.3 Member Function Documentation

**6.486.3.1** `virtual void decaf::util::concurrent::locks::Lock::lock () throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Acquires the lock. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p. 2377) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2377) implementation.

#### Exceptions:

***RuntimeException*** if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3185).

#### 6.486.3.2 virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [pure virtual]

Acquires the lock unless the current thread is interrupted. Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

#### Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p. 2377) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2377) implementation.

#### Exceptions:

***RuntimeException*** if an error occurs while acquiring the lock.

***InterruptedException*** if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3185).

#### 6.486.3.3 virtual Condition\* decaf::util::concurrent::locks::Lock::newCondition ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]

Returns a new **Condition** (p. 1249) instance that is bound to this **Lock** (p. 2377) instance. Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()**

(p. 1251) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

#### Implementation Considerations

The exact operation of the **Condition** (p. 1249) instance depends on the **Lock** (p. 2377) implementation and must be documented by that implementation.

#### Returns:

A new **Condition** (p. 1249) instance for this **Lock** (p. 2377) instance the caller must delete the returned **Condition** (p. 1249) object when done with it.

#### Exceptions:

***RuntimeException*** if an error occurs while creating the **Condition** (p. 1249).

***UnsupportedOperationException*** if this **Lock** (p. 2377) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3186).

**6.486.3.4 virtual bool decaf::util::concurrent::locks::Lock::tryLock**  
**(long long time, const TimeUnit & unit) throw**  
**( decaf::lang::exceptions::RuntimeException, de-**  
**caf::lang::exceptions::InterruptedException ) [pure**  
**virtual]**

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted. If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or \* The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 2377) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2377) implementation.

**Parameters:**

*time* the maximum time to wait for the lock

*unit* the time unit of the time argument

**Returns:**

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

**Exceptions:**

*RuntimeException* if an error occurs while acquiring the lock.

*InterruptedException* if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3186).

### 6.486.3.5 virtual bool decaf::util::concurrent::locks::Lock::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Acquires the lock only if it is free at the time of invocation. Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

```
Lock (p. 2377) lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }
```

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

**Returns:**

true if the lock was acquired and false otherwise

**Exceptions:**

*RuntimeException* if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3187).

### 6.486.3.6 virtual void decaf::util::concurrent::locks::Lock::unlock () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]

Releases the lock. Implementation Considerations

A **Lock** (p. 2377) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 2377) implementation.

**Exceptions:**

*RuntimeException* if an error occurs while acquiring the lock.

***IllegalMonitorStateException*** if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3188).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/`**Lock.h**

## 6.487 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating **locks** (p.174) and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

### Public Member Functions

- `~LockSupport ()`

### Static Public Member Functions

- static void **unpark** (decaf::lang::Thread \*thread) throw ()  
*Makes available the permit for the given thread, if it was not already available.*
- static void **park** () throw ()  
*Disables the current thread for thread scheduling purposes unless the permit is available.*
- static void **parkNanos** (long long nanos) throw ()  
*Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.*
- static void **parkUntil** (long long deadline) throw ()  
*Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.*

### 6.487.1 Detailed Description

Basic thread blocking primitives for creating **locks** (p.174) and other synchronization classes. This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p.3341) class). A call to park will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to unpark makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods park and unpark provide efficient means of blocking and unblocking threads. Races between one thread invoking park and another thread trying to unpark it will preserve liveness, due to the permit. Additionally, park will return if the caller's thread was interrupted, and timeout versions are supported. The park method may also return at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense park serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an unpark to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The park method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither `canProceed` nor any other actions prior to the call to `park` entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of `park` could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
void lock() {
bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add( current );
// Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
!locked.compareAndSet( false, true ) ) {
LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting wasInter-
rupted = true; }
waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt(); }
void unlock() { locked.set( false ); LockSupport.unpark (p. 2385)( waiters.peek() ); } };
```

**Since:**

1.0

## 6.487.2 Constructor & Destructor Documentation

**6.487.2.1** `decaf::util::concurrent::locks::LockSupport::~~LockSupport ()`

## 6.487.3 Member Function Documentation

**6.487.3.1** `static void decaf::util::concurrent::locks::LockSupport::park () throw ()`  
[static]

Disables the current thread for thread scheduling purposes unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes `unpark` with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

**6.487.3.2** `static void decaf::util::concurrent::locks::LockSupport::parkNanos (long long nanos) throw ()` [static]

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:



\* Some other thread invokes `unpark` with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses; or \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

**Parameters:**

*nanos* the maximum number of nanoseconds to wait

**6.487.3.3 static void decaf::util::concurrent::locks::LockSupport::parkUntil (long long *deadline*) throw () [static]**

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

\* Some other thread invokes `unpark` with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The specified deadline passes; or \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

**Parameters:**

*deadline* the absolute time, in milliseconds from the Epoch, to wait until

**6.487.3.4 static void decaf::util::concurrent::locks::LockSupport::unpark (decaf::lang::Thread \* *thread*) throw () [static]**

Makes available the permit for the given thread, if it was not already available. If the thread was blocked on `park` then it will unblock. Otherwise, its next call to `park` is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

**Parameters:**

*thread* the thread to unpark, or NULL in which case the method has no effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/LockSupport.h`

## 6.488 decaf::util::logging::Logger Class Reference

A **Logger** (p. 2386) object is used to log messages for a specific system or application component.

```
#include <src/main/decaf/util/logging/Logger.h>
```

### Public Member Functions

- virtual **~Logger** ()
- const std::string & **getName** () const  
*Gets the name of this **Logger** (p. 2386).*
- **Logger** \* **getParent** () const  
*Gets the parent of this **Logger** (p. 2386) which will be the nearest existing **Logger** (p. 2386) in this Loggers namespace.*
- void **setParent** (**Logger** \*parent)  
*Set (p. 3439) the parent for this **Logger** (p. 2386).*
- void **addHandler** (**Handler** \*handler) throw ( lang::exceptions::NullPointerException )  
*Add a log **Handler** (p. 1978) to receive **logging** (p. 175) messages.*
- void **removeHandler** (**Handler** \*handler)  
*Removes the specified **Handler** (p. 1978) from this logger, ownership of the **Handler** (p. 1978) pointer is returned to the caller.*
- const std::list< **Handler** \* > & **getHandlers** () const  
*Gets a vector containing all the handlers that this class has been assigned to use.*
- void **setFilter** (**Filter** \*filter)  
*Set (p. 3439) a filter to control output on this **Logger** (p. 2386).*
- const **Filter** \* **getFilter** () const  
*Gets the **Filter** (p. 1893) object that this class is using.*
- **Level** **getLevel** () const  
*Get the log **Level** (p. 2332) that has been specified for this **Logger** (p. 2386).*
- void **setLevel** (const **Level** &level)  
*Set (p. 3439) the log level specifying which message levels will be logged by this logger.*
- bool **getUseParentHandlers** () const  
*Discover whether or not this logger is sending its output to its parent logger.*
- void **setUseParentHandlers** (bool value)  
*Specify whether or not this logger should send its output to it's parent **Logger** (p. 2386).*
- virtual void **entering** (const std::string &blockName, const std::string &file, const int line)  
*Logs an **Block Enter** message.*

- virtual void **exiting** (const std::string &blockName, const std::string &file, const int line)  
*Logs an Block Exit message.*
- virtual void **severe** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a SEVERE Level (p. 2332) Log.*
- virtual void **warning** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a WARN Level (p. 2332) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a INFO Level (p. 2332) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a DEBUG Level (p. 2332) Log.*
- virtual void **config** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a CONFIG Level (p. 2332) Log.*
- virtual void **fine** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINE Level (p. 2332) Log.*
- virtual void **finer** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINER Level (p. 2332) Log.*
- virtual void **finest** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINEST Level (p. 2332) Log.*
- virtual void **throwing** (const std::string &file, const int line, const std::string functionName, const decaf::lang::Throwable &thrown)  
*Log throwing an exception.*
- virtual bool **isLoggable** (const Level &level) const  
*Check if a message of the given level would actually be logged by this logger.*
- virtual void **log** (LogRecord &record)  
*Log a LogRecord (p. 2413).*
- virtual void **log** (const Level &level, const std::string &message)  
*Log a message, with no arguments.*
- virtual void **log** (const Level &levels, const std::string &file, const int line, const std::string &message,...)  
*Log a message, with the list of params that is formatted into the message string.*

- virtual void **log** (const **Level** &level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)

*Log a message, with associated Throwable information.*

## Static Public Member Functions

- static **Logger** \* **getAnonymousLogger** ()  
*Creates an anonymous logger.*
- static **Logger** \* **getLogger** (const std::string &name)  
*Find or create a logger for a named subsystem.*

## Protected Member Functions

- **Logger** (const std::string &name)  
*Creates a new instance of the **Logger** (p. 2386) with the given name.*

### 6.488.1 Detailed Description

A **Logger** (p. 2386) object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 2386) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 160) or **org.apache.decaf**. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 2386) namespace.

**Logger** (p. 2386) objects may be obtained by calls on one of the **getLogger** factory methods. These will either create a new **Logger** (p. 2386) or return a suitable existing **Logger** (p. 2386).

Logging messages will be forwarded to registered **Handler** (p. 1978) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 2386) keeps track of a "parent" **Logger** (p. 2386), which is its nearest existing ancestor in the **Logger** (p. 2386) namespace.

Each **Logger** (p. 2386) has a "Level" associated with it. This reflects a minimum **Level** (p. 2332) that this logger cares about. If a Logger's level is set to **Level::INHERIT** (p. 2336), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the **logging** (p. 175) configuration file, as described in the description of the **LogManager** (p. 2406) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 2395) method. If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each **logging** (p. 175) call the **Logger** (p. 2386) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the **logging** (p. 175) call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 2386) will allocate a **LogRecord** (p. 2413) to describe the **logging** (p. 175) message. It will then call a **Filter** (p. 1893) (if present) to do a

more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 2413) to its output **Handlers**. By default, loggers also publish to their parent's **Handlers**, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 1978), which will typically call a **Formatter** (p. 1964).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 2413) is actually written to an external sink.

All methods on **Logger** (p. 2386) are thread safe.

#### Since:

1.0

## 6.488.2 Constructor & Destructor Documentation

### 6.488.2.1 decaf::util::logging::Logger::Logger (const std::string & *name*) [protected]

Creates a new instance of the **Logger** (p. 2386) with the given name. The logger will be initially configured with a null **Level** (p. 2332) and with useParentHandlers true.

#### Parameters:

*name* A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as **decaf.net** (p. 160) or org.apache.decaf. It may be empty for anonymous Loggers.

### 6.488.2.2 virtual decaf::util::logging::Logger::~~Logger () [virtual]

## 6.488.3 Member Function Documentation

### 6.488.3.1 void decaf::util::logging::Logger::addHandler (Handler \* *handler*) throw (lang::exceptions::NullPointerException )

Add a log **Handler** (p. 1978) to receive **logging** (p. 175) messages. By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 2386) is configured with a set of **Handlers** that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 1978) is passed to the **Logger** (p. 2386) and the **Handler** (p. 1978) will be deleted when this **Logger** (p. 2386) is destroyed unless the caller first calls **removeHandler** with the same pointer value as was originally given.

#### Parameters:

*handler* A Logging **Handler** (p. 1978)

#### Exceptions:

*NullPointerException* if the **Handler** (p. 1978) given is NULL.

**6.488.3.2** `virtual void decaf::util::logging::Logger::config (const std::string & file,  
const int line, const std::string functionName, const std::string &  
message)` [virtual]

Log a CONFIG **Level** (p. 2332) Log. If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects.

**Parameters:**

*file* The file name where the log was generated.  
*line* The line number where the log was generated.  
*functionName* The name of the function that logged this.  
*message* The message to log at this **Level** (p. 2332).

**6.488.3.3** `virtual void decaf::util::logging::Logger::debug (const std::string & file,  
const int line, const std::string functionName, const std::string &  
message)` [virtual]

Log a DEBUG **Level** (p. 2332) Log. If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects.

**Parameters:**

*file* The file name where the log was generated.  
*line* The line number where the log was generated.  
*functionName* The name of the function that logged this.  
*message* The message to log at this **Level** (p. 2332).

**6.488.3.4** `virtual void decaf::util::logging::Logger::entering (const std::string &  
blockName, const std::string & file, const int line)` [virtual]

Logs an Block Enter message. This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2335) log level.

**Parameters:**

*blockName* The source block name, (usually ClassName::MethodName, or MethodName).  
*file* The source file name where this method was called from.  
*line* The source line number where this method was called from.

**6.488.3.5** `virtual void decaf::util::logging::Logger::exiting (const std::string &  
blockName, const std::string & file, const int line)` [virtual]

Logs an Block Exit message. This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2335) log level.

**Parameters:**

*blockName* The source block name, (usually ClassName::MethodName, or MethodName).  
*file* The source file name where this method was called from.  
*line* The source line number where this method was called from.

**6.488.3.6** virtual void decaf::util::logging::Logger::fine (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a FINE **Level** (p.2332) Log. If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output **Handler** (p.1978) objects.

**Parameters:**

*file* The file name where the log was generated.  
*line* The line number where the log was generated.  
*functionName* The name of the function that logged this.  
*message* The message to log at this **Level** (p.2332).

**6.488.3.7** virtual void decaf::util::logging::Logger::finer (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a FINER **Level** (p.2332) Log. If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output **Handler** (p.1978) objects.

**Parameters:**

*file* The file name where the log was generated.  
*line* The line number where the log was generated.  
*functionName* The name of the function that logged this.  
*message* The message to log at this **Level** (p.2332).

**6.488.3.8** virtual void decaf::util::logging::Logger::finest (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a FINEST **Level** (p.2332) Log. If the logger is currently enabled for the FINEST message level then the given message is forwarded to all the registered output **Handler** (p.1978) objects.

**Parameters:**

*file* The file name where the log was generated.  
*line* The line number where the log was generated.  
*functionName* The name of the function that logged this.  
*message* The message to log at this **Level** (p.2332).

**6.488.3.9** static Logger\* decaf::util::logging::Logger::getAnonymousLogger () [static]

Creates an anonymous logger. The newly created **Logger** (p.2386) is not registered in the **Log-Manager** (p.2406) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

**Returns:**

Newly created anonymous logger

**6.488.3.10** `const Filter* decaf::util::logging::Logger::getFilter () const [inline]`

Gets the **Filter** (p. 1893) object that this class is using.

**Returns:**

the **Filter** (p. 1893) in use, (can be NULL).

**6.488.3.11** `const std::list<Handler*>& decaf::util::logging::Logger::getHandlers () const`

Gets a vector containing all the handlers that this class has been assigned to use.

**Returns:**

a list of handlers that are used by this logger

**6.488.3.12** `Level decaf::util::logging::Logger::getLevel () const [inline]`

Get the log **Level** (p. 2332) that has been specified for this **Logger** (p. 2386). The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

**Returns:**

the level that is currently set

**6.488.3.13** `static Logger* decaf::util::logging::Logger::getLogger (const std::string & name) [static]`

Find or create a logger for a named subsystem. If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 2406) and it will be configured to also send **logging** (p. 175) output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 2406) global namespace.

**Parameters:**

**name** - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as **cms** (p. 140) or **activemq.core.ActiveMQConnection** (p. 273)

**Returns:**

a suitable logger.



**6.488.3.14** `const std::string& decaf::util::logging::Logger::getName () const [inline]`

Gets the name of this **Logger** (p. 2386).

**Returns:**

logger name

**6.488.3.15** `Logger* decaf::util::logging::Logger::getParent () const [inline]`

Gets the parent of this **Logger** (p. 2386) which will be the nearest existing **Logger** (p. 2386) in this Loggers namespace. If this is the Root **Logger** (p. 2386) than this method returns NULL.

**Returns:**

Pointer to this Loggers nearest parent **Logger** (p. 2386).

**6.488.3.16** `bool decaf::util::logging::Logger::getUseParentHandlers () const [inline]`

Discover whether or not this logger is sending its output to its parent logger.

**Returns:**

true if using Parent Handlers

**6.488.3.17** `virtual void decaf::util::logging::Logger::info (const std::string & file, const int line, const std::string functionName, const std::string & message) [virtual]`

Log a INFO **Level** (p. 2332) Log. If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects.

**Parameters:**

*file* The file name where the log was generated.

*line* The line number where the log was generated.

*functionName* The name of the function that logged this.

*message* The message to log at this **Level** (p. 2332).

**6.488.3.18** `virtual bool decaf::util::logging::Logger::isLoggable (const Level & level) const [virtual]`

Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.

**Parameters:**

*level* - a message **logging** (p. 175) level

**Returns:**

true if the given message level is currently being logged.

**6.488.3.19** `virtual void decaf::util::logging::Logger::log (const Level & level,  
const std::string & file, const int line, const std::string & message,  
lang::Exception & ex)` [virtual]

Log a message, with associated Throwable information. If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2413) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 2413) thrown property, rather than the **LogRecord** (p. 2413) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 2413) message property.

**Parameters:**

*level* the **Level** (p. 2332) to log at.

*file* File that the message was logged in.

*line* the line number where the message was logged at.

*message* the message to log.

*ex* the Exception to log

**6.488.3.20** `virtual void decaf::util::logging::Logger::log (const Level & levels, const  
std::string & file, const int line, const std::string & message, ...)`  
[virtual]

Log a message, with the list of params that is formatted into the message string. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects

**Parameters:**

*level* the **Level** (p. 2332) to log at

*file* the message to log

*line* the line in the file

... variable length argument to format the message string.

**6.488.3.21** `virtual void decaf::util::logging::Logger::log (const Level & level, const  
std::string & message)` [virtual]

Log a message, with no arguments. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects

**Parameters:**

*level* the **Level** (p. 2332) to log at

*message* the message to log

**6.488.3.22 virtual void decaf::util::logging::Logger::log (LogRecord & *record*)**  
[virtual]

Log a **LogRecord** (p. 2413). All the other **logging** (p. 175) methods in this class call through this method to actually perform any **logging** (p. 175). Subclasses can override this single method to capture all log activity.

**Parameters:**

*record* - the **LogRecord** (p. 2413) to be published

**6.488.3.23 void decaf::util::logging::Logger::removeHandler (Handler \* *handler*)**

Removes the specified **Handler** (p. 1978) from this logger, ownership of the **Handler** (p. 1978) pointer is returned to the caller. Returns silently if the given **Handler** (p. 1978) is not found.

**Parameters:**

*handler* The **Handler** (p. 1978) to remove

**6.488.3.24 void decaf::util::logging::Logger::setFilter (Filter \* *filter*)**

**Set** (p. 3439) a filter to control output on this **Logger** (p. 2386). After passing the initial "level" check, the **Logger** (p. 2386) will call this **Filter** (p. 1893) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

**Parameters:**

*filter* The **Filter** (p. 1893) to use, (can be NULL).

**6.488.3.25 void decaf::util::logging::Logger::setLevel (const Level & *level*)** [inline]

**Set** (p. 3439) the log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded. The level value **Level::OFF** (p. 2336) can be used to turn off **logging** (p. 175).

If the new level is the **INHERIT Level** (p. 2332), it means that this node should inherit its level from its nearest ancestor with a specific (non-**INHERIT**) level value.

**Parameters:**

*level* The new **Level** (p. 2332) value to use when **logging** (p. 175).

**6.488.3.26 void decaf::util::logging::Logger::setParent (Logger \* *parent*)** [inline]

**Set** (p. 3439) the parent for this **Logger** (p. 2386). This method is used by the **LogManager** (p. 2406) to update a **Logger** (p. 2386) when the namespace changes.

It should not be called from application **code** (p. 1183).

**6.488.3.27** `void decaf::util::logging::Logger::setUseParentHandlers (bool value)`  
[inline]

Specify whether or not this logger should send its output to its parent **Logger** (p. 2386). This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

**Parameters:**

*value* True is output is to be written to the parent

**6.488.3.28** `virtual void decaf::util::logging::Logger::severe (const std::string & file,  
const int line, const std::string functionName, const std::string &  
message)` [virtual]

Log a SEVERE **Level** (p. 2332) Log. If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects.

**Parameters:**

*file* The file name where the log was generated.

*line* The line number where the log was generated.

*functionName* The name of the function that logged this.

*message* The message to log at this **Level** (p. 2332).

**6.488.3.29** `virtual void decaf::util::logging::Logger::throwing (const std::string  
& file, const int line, const std::string functionName, const  
decaf::lang::Throwable & thrown)` [virtual]

Log throwing an exception. This is a convenience method to log that a method is terminating by throwing an exception. The **logging** (p. 175) is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2413) which is forwarded to all registered output handlers. The LogRecord's message is set to "THROW".

**Parameters:**

*file* The file name where the log was generated.

*line* The line number where the log was generated.

*functionName* The name of the function that logged this.

*thrown* The Throwable that will be thrown, will be cloned.

**6.488.3.30** `virtual void decaf::util::logging::Logger::warning (const std::string &  
file, const int line, const std::string functionName, const std::string &  
message)` [virtual]

Log a WARN **Level** (p. 2332) Log. If the logger is currently enabled for the WARN message level then the given message is forwarded to all the registered output **Handler** (p. 1978) objects.

**Parameters:**

- file* The file name where the log was generated.
- line* The line number where the log was generated.
- functionName* The name of the function that logged this.
- message* The message to log at this **Level** (p. 2332).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Logger.h**

## 6.489 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

### Public Member Functions

- **LoggerHierarchy** ()
- virtual **~LoggerHierarchy** ()

### 6.489.1 Constructor & Destructor Documentation

**6.489.1.1 decaf::util::logging::LoggerHierarchy::LoggerHierarchy ()**

**6.489.1.2 virtual decaf::util::logging::LoggerHierarchy::~~LoggerHierarchy ()**  
[virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LoggerHierarchy.h**

## 6.490 activemq::io::LoggingInputStream Class Reference

#include <src/main/activemq/io/LoggingInputStream.h> Inheritance diagram for activemq::io::LoggingInputStream:

### Public Member Functions

- **LoggingInputStream** (decaf::io::InputStream \*inputStream, bool own=false)  
*Creates a DataInputStream that uses the specified underlying InputStream.*
- virtual ~**LoggingInputStream** ()

### Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.490.1 Constructor & Destructor Documentation

#### 6.490.1.1 activemq::io::LoggingInputStream::LoggingInputStream (decaf::io::InputStream \* inputStream, bool own = false)

Creates a DataInputStream that uses the specified underlying InputStream.

#### Parameters:

*inputStream* the InputStream instance to wrap.

*own* indicates if this class owns the wrapped string defaults to false.

#### 6.490.1.2 virtual activemq::io::LoggingInputStream::~~LoggingInputStream () [virtual]

### 6.490.2 Member Function Documentation

#### 6.490.2.1 virtual int activemq::io::LoggingInputStream::doReadArrayBounded (unsigned char \* buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from decaf::io::FilterInputStream (p.1897).

**6.490.2.2** `virtual int activemq::io::LoggingInputStream::doReadByte () throw (decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1897).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`



## 6.491 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

#include <src/main/activemq/io/LoggingOutputStream.h> Inheritance diagram for activemq::io::LoggingOutputStream:

### Public Member Functions

- **LoggingOutputStream** (OutputStream \*next, bool **own**=false)  
*Constructor.*
- virtual ~**LoggingOutputStream** ()

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.491.1 Detailed Description

OutputStream filter that just logs the data being written.

### 6.491.2 Constructor & Destructor Documentation

#### 6.491.2.1 activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream \* next, bool own = false)

Constructor.

#### Parameters:

- next* The OutputStream to wrap an write logs to.
- own* If true, this object will control the lifetime of the output stream that it encapsulates.

#### 6.491.2.2 virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream () [virtual]

### 6.491.3 Member Function Documentation

#### 6.491.3.1 virtual void activemq::io::LoggingOutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from decaf::io::FilterOutputStream (p. 1902).

**6.491.3.2** `virtual void activemq::io::LoggingOutputStream::doWriteByte (unsigned char c) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1902).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

## 6.492 activemq::transport::logging::LoggingTransport Class Reference

A **transport** (p. 97) filter that logs **commands** (p. 87) as they are sent/received.

#include <src/main/activemq/transport/logging/LoggingTransport.h> Inheritance diagram for activemq::transport::logging::LoggingTransport:

### Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)

*Constructor.*

- virtual ~**LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

*Event handler for the receipt of a command.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

### 6.492.1 Detailed Description

A **transport** (p. 97) filter that logs **commands** (p. 87) as they are sent/received.

### 6.492.2 Constructor & Destructor Documentation

#### 6.492.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport (const **Pointer**< **Transport** > & next)

Constructor.

#### Parameters:

*next* - the next **Transport** (p. 3883) in the chain

**6.492.2.2** `virtual`  
`activemq::transport::logging::LoggingTransport::~~LoggingTransport ()`  
`[inline, virtual]`

### 6.492.3 Member Function Documentation

**6.492.3.1** `virtual void activemq::transport::logging::LoggingTransport::onCommand`  
`(const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

#### Parameters:

*command* - the received command object.

Reimplemented from `activemq::transport::TransportFilter` (p. 3895).

**6.492.3.2** `virtual void activemq::transport::logging::LoggingTransport::oneway`  
`(const Pointer< Command > & command) throw ( decaf::io::IOException,`  
`decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

#### Parameters:

*command* the command to be sent.

#### Exceptions:

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this `transport` (p. 97).

Reimplemented from `activemq::transport::TransportFilter` (p. 3895).

**6.492.3.3** `virtual Pointer<Response> ac-`  
`tivemq::transport::logging::LoggingTransport::request`  
`(const Pointer< Command > & command, un-`  
`signed int timeout) throw ( decaf::io::IOException,`  
`decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

#### Parameters:

*command* the command that is sent as a request

*timeout* the time to wait for a response.

#### Exceptions:

*UnsupportedOperationException*.

Reimplemented from `activemq::transport::TransportFilter` (p. 3896).

**6.492.3.4** virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & *command*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Not supported by this class - throws an exception.

**Parameters:**

*command* the command that is sent as a request

**Exceptions:**

*UnsupportedOperationException.*

Reimplemented from **activemq::transport::TransportFilter** (p. 3897).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/logging/**LoggingTransport.h**

## 6.493 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p.2406) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

### Public Member Functions

- virtual **~LogManager** ()
- bool **addLogger** (**Logger** \*logger) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Add a named logger.*
- **Logger** \* **getLogger** (const std::string &name)  
*Retrieves or creates a new **Logger** (p.2386) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- int **getLoggerNames** (const std::vector< std::string > &names)  
*Gets a list of known **Logger** (p.2386) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.*
- void **setProperties** (const **util::Properties** &properties)  
*Sets the **Properties** (p.3126) this **LogManager** (p.2406) should use to configure its loggers.*
- const **util::Properties** & **getProperties** () const  
*Gets a reference to the Logging **Properties** (p.3126) used by this logger.*
- std::string **getProperty** (const std::string &name)  
*Gets the value of a named property of this **LogManager** (p.2406).*
- void **addPropertyChangeListener** (PropertyChangeListener \*listener)  
*Adds a change listener for **LogManager** (p.2406) **Properties** (p.3126), adding the same instance of a change event listener does nothing.*
- void **removePropertyChangeListener** (PropertyChangeListener \*listener)  
*Removes a properties change listener from the **LogManager** (p.2406), if the listener is not found of the param is NULL this method returns silently.*
- void **readConfiguration** () throw ( decaf::io::IOException )  
*Reinitialize the **logging** (p.175) properties and reread the **logging** (p.175) configuration.*
- void **readConfiguration** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reinitialize the **logging** (p.175) properties and reread the **logging** (p.175) configuration from the given stream, which should be in **decaf.util.Properties** (p.3126) format.*
- void **reset** ()  
*Reset the **logging** (p.175) configuration.*

## Static Public Member Functions

- static **LogManager** & **getLogManager** ()  
*Get the global **LogManager** (p. 2406) instance.*

## Protected Member Functions

- **LogManager** ()  
*Constructor, hidden to protect against direct instantiation.*
- **LogManager** (const **LogManager** &manager)  
*Copy Constructor.*
- void **operator=** (const **LogManager** &manager)  
*Assignment operator.*

## Friends

- class **decaf::lang::Runtime**

### 6.493.1 Detailed Description

There is a single global **LogManager** (p. 2406) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p. 2406) object:

\* Manages a hierarchical namespace of **Logger** (p. 2386) objects. All named Loggers are stored in this namespace. \* Manages a set of **logging** (p. 175) control properties. These are simple key-value pairs that can be used by Handlers and other **logging** (p. 175) objects to configure themselves.

The global **LogManager** (p. 2406) object can be retrieved using **LogManager::getLogManager()** (p. 2410). The **LogManager** (p. 2406) object is created during class initialization and cannot subsequently be changed.

\*\*\*TODO\*\*\* By default, the **LogManager** (p. 2406) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default **logging** (p. 175) configuration for all uses of that JRE.

In addition, the **LogManager** (p. 2406) uses two optional system properties that allow more control over reading the initial configuration:

\* "decaf.logger.config.class" \* "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI\_CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the **LogManager** (p. 2406).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 3126) format). The initial **logging** (p. 175) configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 2406) will read its initial configuration from a properties file "lib/logging.properties" in the working directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. \*\*\*TODO\*\*\*

The global **logging** (p. 175) properties may include:

- \* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 2386) (the **Logger** (p. 2386) named ""). Each class name must be for a **Handler** (p. 1978) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- \* A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1978) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- \* A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the **logging** (p. 175) message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1978) needs to be configured for this logger otherwise no **logging** (p. 175) messages are delivered.
- \* A property "config". This property is intended to allow arbitrary configuration **code** (p. 1183) to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary **code** (p. 1183) to update the **logging** (p. 175) configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and a.b2 are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 2406) object are multi-thread safe.

Since:

1.0

## 6.493.2 Constructor & Destructor Documentation

**6.493.2.1** `virtual decaf::util::logging::LogManager::~~LogManager ()` [virtual]

**6.493.2.2** `decaf::util::logging::LogManager::LogManager ()` [protected]

Constructor, hidden to protect against direct instantiation.

**6.493.2.3** `decaf::util::logging::LogManager::LogManager (const LogManager & manager)` [protected]

Copy Constructor.



**Parameters:**

*manager* the Manager to copy

**6.493.3 Member Function Documentation**

**6.493.3.1** `bool decaf::util::logging::LogManager::addLogger (Logger *  
logger) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException )`

Add a named logger. This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 2386) factory methods call this method to register each newly created **Logger** (p. 2386).

**Parameters:**

*logger* The new **Logger** (p. 2386) instance to add to this **LogManager** (p. 2406).

**Exceptions:**

*NullPointerException* if logger is NULL.

*IllegalArgumentException* if the logger has no name.

**6.493.3.2** `void decaf::util::logging::LogManager::addPropertyChangeListener  
(PropertyChangeListener * listener)`

Adds a change listener for **LogManager** (p. 2406) **Properties** (p. 3126), adding the same instance of a change event listener does nothing.

**Parameters:**

*listener* The PropertyChangeListener to add (can be NULL).

**6.493.3.3** `Logger* decaf::util::logging::LogManager::getLogger (const std::string &  
name)`

Retrieves or creates a new **Logger** (p. 2386) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

**Parameters:**

*name* The name of the **Logger** (p. 2386).

**6.493.3.4** `int decaf::util::logging::LogManager::getLoggerNames (const std::vector<  
std::string > & names)`

Gets a list of known **Logger** (p. 2386) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.

**Parameters:**

*names* STL Vector to hold string logger names.

**Returns:**

names count of how many loggers were inserted.

**6.493.3.5** `static LogManager& decaf::util::logging::LogManager::getLogManager ()`  
[static]

Get the global **LogManager** (p. 2406) instance.

**Returns:**

A reference to the global **LogManager** (p. 2406) instance.

**6.493.3.6** `const util::Properties& decaf::util::logging::LogManager::getProperties ()`  
`const` [inline]

Gets a reference to the Logging **Properties** (p. 3126) used by this logger.

**Returns:**

The **Logger** (p. 2386) **Properties** (p. 3126) Pointer

**6.493.3.7** `std::string decaf::util::logging::LogManager::getProperty (const`  
`std::string & name)`

Gets the value of a named property of this **LogManager** (p. 2406).

**Parameters:**

*name* The name of the Property to retrieve.

**Returns:**

the value of the property

**6.493.3.8** `void decaf::util::logging::LogManager::operator= (const LogManager &`  
`manager)` [protected]

Assignment operator.

**Parameters:**

*manager* the manager to assign from

### 6.493.3.9 void decaf::util::logging::LogManager::readConfiguration (decaf::io::InputStream \* *stream*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

Reinitialize the **logging** (p.175) properties and reread the **logging** (p.175) configuration from the given stream, which should be in **decaf.util.Properties** (p.3126) format. A **PropertyChangeEvent** will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p.2395), if the target **Logger** (p.2386) exists.

#### Parameters:

*stream* The **InputStream** to read the **Properties** (p.3126) from.

#### Exceptions:

**NullPointerException** if stream is NULL.

**IOException** if an I/O error occurs.

### 6.493.3.10 void decaf::util::logging::LogManager::readConfiguration () throw ( decaf::io::IOException )

Reinitialize the **logging** (p.175) properties and reread the **logging** (p.175) configuration. The same rules are used for locating the configuration properties as are used at startup. So normally the **logging** (p.175) properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p.2395), if the target **Logger** (p.2386) exists.

A **PropertyChangeEvent** will be fired after the properties are read.

#### Exceptions:

**IOException** if an I/O error occurs.

### 6.493.3.11 void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener \* *listener*)

Removes a properties change listener from the **LogManager** (p.2406), if the listener is not found of the param is NULL this method returns silently.

#### Parameters:

*listener* The **PropertyChangeListener** to remove from the listeners set.

### 6.493.3.12 void decaf::util::logging::LogManager::reset ()

Reset the **logging** (p.175) configuration. For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to **INHERIT**. The root logger's level is set to **Level::INFO** (p.2336).

#### 6.493.3.13 void decaf::util::logging::LogManager::setProperties (const util::Properties & *properties*)

Sets the **Properties** (p. 3126) this **LogManager** (p. 2406) should use to configure its loggers. Once set a properties change event is fired.

##### Parameters:

*properties* Pointer to read the configuration from

### 6.493.4 Friends And Related Function Documentation

#### 6.493.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

## 6.494 decaf::util::logging::LogRecord Class Reference

**LogRecord** (p. 2413) objects are used to pass **logging** (p. 175) requests between the **logging** (p. 175) framework and individual log Handlers.

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

### Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const  
*Get **Level** (p. 2332) of this log record.*
- void **setLevel** (**Level** value)  
*Set (p. 3439) the **Level** (p. 2332) of this Log Record.*
- const std::string & **getLoggerName** () const  
*Gets the Source Logger's Name.*
- void **setLoggerName** (const std::string &loggerName)  
*Sets the Source Logger's Name.*
- const std::string & **getSourceFile** () const  
*Gets the Source Log File name.*
- void **setSourceFile** (const std::string &sourceFile)  
*Sets the Source Log File Name.*
- unsigned int **getSourceLine** () const  
*Gets the Source Log line number.*
- void **setSourceLine** (unsigned int sourceLine)  
*Sets the Source Log line number.*
- const std::string & **getMessage** () const  
*Gets the Message to be Logged.*
- void **setMessage** (const std::string &message)  
*Sets the Message to be Logged.*
- const std::string & **getSourceFunction** () const  
*Gets the name of the function where this log was logged.*
- void **setSourceFunction** (const std::string &functionName)  
*Sets the name of the function where this log was logged.*
- long long **getTimestamp** () const  
*Gets the time in mills that this message was logged.*

- void **setTimestamp** (long long timeStamp)  
*Sets the time in mills that this message was logged.*
- long long **getTreadId** () const  
*Gets the Thread Id where this Log was created.*
- void **setTreadId** (long long threadId)  
*Sets the Thread Id where this Log was created.*
- **decaf::lang::Throwable \* getThrown** () const  
*Gets any Throwable associated with this **LogRecord** (p. 2413).*
- void **setThrown** (**decaf::lang::Throwable \*thrown**)  
*Sets the Throwable associated with this **LogRecord** (p. 2413), the pointer becomes the property of this instance of the **LogRecord** (p. 2413) and will be deleted when the record is destroyed.*

### 6.494.1 Detailed Description

**LogRecord** (p. 2413) objects are used to pass **logging** (p. 175) requests between the **logging** (p. 175) framework and individual log Handlers. When a **LogRecord** (p. 2413) is passed into the **logging** (p. 175) framework it logically belongs to the framework and should no longer be used or updated by the client application.

**Since:**

1.0

### 6.494.2 Constructor & Destructor Documentation

**6.494.2.1 decaf::util::logging::LogRecord::LogRecord ()**

**6.494.2.2 virtual decaf::util::logging::LogRecord::~~LogRecord ()** [virtual]

### 6.494.3 Member Function Documentation

**6.494.3.1 Level decaf::util::logging::LogRecord::getLevel () const** [inline]

Get **Level** (p. 2332) of this log record.

**Returns:**

**Level** (p. 2332) enumeration value.

**6.494.3.2 const std::string& decaf::util::logging::LogRecord::getLoggerName () const** [inline]

Gets the Source Logger's Name.

**Returns:**

the source loggers name

**6.494.3.3** `const std::string& decaf::util::logging::LogRecord::getMessage () const [inline]`

Gets the Message to be Logged.

**Returns:**

the source logger's message

**6.494.3.4** `const std::string& decaf::util::logging::LogRecord::getSourceFile () const [inline]`

Gets the Source Log File name.

**Returns:**

the source loggers name

**6.494.3.5** `const std::string& decaf::util::logging::LogRecord::getSourceFunction () const [inline]`

Gets the name of the function where this log was logged.

**Returns:**

the source logger's message

**6.494.3.6** `unsigned int decaf::util::logging::LogRecord::getSourceLine () const [inline]`

Gets the Source Log line number.

**Returns:**

the source loggers line number

**6.494.3.7** `decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown () const [inline]`

Gets any Throwable associated with this **LogRecord** (p. 2413).

**Returns:**

point to a Throwable instance or Null.

**6.494.3.8** `long long decaf::util::logging::LogRecord::getTimestamp () const [inline]`

Gets the time in mills that this message was logged.

**Returns:**

UTC time in milliseconds

**6.494.3.9** `long long decaf::util::logging::LogRecord::getTreadId () const [inline]`

Gets the Thread Id where this Log was created.

**Returns:**

the source loggers line number

**6.494.3.10** `void decaf::util::logging::LogRecord::setLevel (Level value) [inline]`

Set (p. 3439) the **Level** (p. 2332) of this Log Record.

**Parameters:**

*value* **Level** (p. 2332) Enumeration Value

**6.494.3.11** `void decaf::util::logging::LogRecord::setLoggerName (const std::string & loggerName) [inline]`

Sets the Source Logger's Name.

**Parameters:**

*loggerName* the source loggers name

**6.494.3.12** `void decaf::util::logging::LogRecord::setMessage (const std::string & message) [inline]`

Sets the Message to be Logged.

**Parameters:**

*message* the source loggers message

**6.494.3.13** `void decaf::util::logging::LogRecord::setSourceFile (const std::string & sourceFile) [inline]`

Sets the Source Log File Name.

**Parameters:**

*sourceFile* the source loggers name

**6.494.3.14** `void decaf::util::logging::LogRecord::setSourceFunction (const std::string & functionName) [inline]`

Sets the name of the function where this log was logged.

**Parameters:**

*functionName* the source of the log



**6.494.3.15** void decaf::util::logging::LogRecord::setSourceLine (unsigned int *sourceLine*) [inline]

Sets the Source Log line number.

**Parameters:**

*sourceLine* the source logger's line number

**6.494.3.16** void decaf::util::logging::LogRecord::setThrown (decaf::lang::Throwable \* *thrown*) [inline]

Sets the Throwable associated with this **LogRecord** (p. 2413), the pointer becomes the property of this instance of the **LogRecord** (p. 2413) and will be deleted when the record is destroyed.

**Parameters:**

*thrown* A pointer to a Throwable that will be associated with this record.

**6.494.3.17** void decaf::util::logging::LogRecord::setTimestamp (long long *timeStamp*) [inline]

Sets the time in mills that this message was logged.

**Parameters:**

*timeStamp* UTC Time in Milliseconds.

**6.494.3.18** void decaf::util::logging::LogRecord::setTreadId (long long *threadId*) [inline]

Sets the Thread Id where this Log was created.

**Parameters:**

*threadId* the source logger's line number

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

## 6.495 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

### Public Member Functions

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)  
*Writes a message to the output destination.*
- virtual void **log** (const std::string &message)  
*Writes a message to the output destination.*

### Static Public Member Functions

- static **LogWriter & getInstance** ()  
*Get the singleton instance.*
- static void **returnInstance** ()  
*Returns a Checked out instance of this Writer.*
- static void **destroy** ()  
*Forcefully Delete the Instance of this **LogWriter** (p. 2418) even if there are outstanding references.*

### 6.495.1 Constructor & Destructor Documentation

6.495.1.1 **decaf::util::logging::LogWriter::LogWriter** ()

6.495.1.2 **virtual decaf::util::logging::LogWriter::~~LogWriter** () [virtual]

### 6.495.2 Member Function Documentation

6.495.2.1 **static void decaf::util::logging::LogWriter::destroy** () [static]

Forcefully Delete the Instance of this **LogWriter** (p. 2418) even if there are outstanding references.

6.495.2.2 **static LogWriter& decaf::util::logging::LogWriter::getInstance** ()  
[static]

Get the singleton instance.

**6.495.2.3** virtual void decaf::util::logging::LogWriter::log (const std::string & *message*) [virtual]

Writes a message to the output destination.

**Parameters:**

*message*

**6.495.2.4** virtual void decaf::util::logging::LogWriter::log (const std::string & *file*, const int *line*, const std::string & *prefix*, const std::string & *message*) [virtual]

Writes a message to the output destination.

**Parameters:**

*file*

*line*

*prefix*

*message*

**6.495.2.5** static void decaf::util::logging::LogWriter::returnInstance () [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogWriter.h**

## 6.496 decaf::lang::Long Class Reference

#include <src/main/decaf/lang/Long.h> Inheritance diagram for decaf::lang::Long:

### Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const  
*Compares this **Long** (p. 2420) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Long** &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const long long &l) const  
*Compares this **Long** (p. 2420) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const long long &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static int **bitCount** (long long value)  
*Returns the number of one-bits in the two's complement binary representation of the specified int value.*
- static **Long decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3665) into a **Long** (p. 2420).*
- static long long **highestOneBit** (long long value)  
*Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*
- static long long **lowestOneBit** (long long value)  
*Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*
- static int **numberOfLeadingZeros** (long long value)  
*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.*
- static int **numberOfTrailingZeros** (long long value)  
*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.*
- static long long **parseLong** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal long.*
- static long long **parseLong** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Long** (p. 2420) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.*
- static long long **reverse** (long long value)  
*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.*
- static long long **rotateLeft** (long long value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.*
- static long long **rotateRight** (long long value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.*
- static int **signum** (long long value)

*Returns the signum function of the specified value.*

- static std::string **toString** (long long value)

*Converts the long to a **String** (p. 3665) representation.*

- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)

*Returns a string representation of the integer argument as an unsigned integer in base 16.*

- static std::string **toOctalString** (long long value)

*Returns a string representation of the long long argument as an unsigned long long in base 8.*

- static std::string **toBinaryString** (long long value)

*Returns a string representation of the long long argument as an unsigned long long in base 2.*

- static **Long** **valueOf** (long long value)

*Returns a **Long** (p. 2420) instance representing the specified int value.*

- static **Long** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )

*Returns a **Long** (p. 2420) object holding the value given by the specified std::string.*

- static **Long** **valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )

*Returns a **Long** (p. 2420) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

## Static Public Attributes

- static const int **SIZE** = 64

*The size in bits of the primitive long long type.*

- static const long long **MAX\_VALUE** = (long long)0x7FFFFFFFFFFFFFFFFFLL

*The maximum value that the primitive type can hold.*

- static const long long **MIN\_VALUE** = (long long)0x8000000000000000LL

*The minimum value that the primitive type can hold.*

## 6.496.1 Constructor & Destructor Documentation

### 6.496.1.1 decaf::lang::Long::Long (long long value)

#### Parameters:

*value* - the primitive long long to wrap

### 6.496.1.2 decaf::lang::Long::Long (const std::string & *value*) throw ( exceptions::NumberFormatException )

#### Parameters:

*value* - the long long formatted string to wrap

#### Exceptions:

*NumberFormatException*

### 6.496.1.3 virtual decaf::lang::Long::~~Long () [inline, virtual]

## 6.496.2 Member Function Documentation

### 6.496.2.1 static int decaf::lang::Long::bitCount (long long *value*) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

#### Parameters:

*value* - the long long to count

#### Returns:

the number of one-bits in the two's complement binary representation of the specified long long value.

### 6.496.2.2 virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2835).

### 6.496.2.3 virtual int decaf::lang::Long::compareTo (const long long & *l*) const [virtual]

Compares this **Long** (p. 2420) instance with another.

#### Parameters:

*l* - the **Integer** (p. 2077) instance to be compared

#### Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< long long > (p. 1214).

**6.496.2.4 virtual int decaf::lang::Long::compareTo (const Long & *l*) const**  
[virtual]

Compares this **Long** (p. 2420) instance with another.

**Parameters:**

*l* - the **Long** (p. 2420) instance to be compared

**Returns:**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.496.2.5 static Long decaf::lang::Long::decode (const std::string & *value*) throw ( exceptions::NumberFormatException )** [static]

Decodes a **String** (p. 3665) into a **Long** (p. 2420). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3665) is the minus sign. No whitespace characters are permitted in the string.

**Parameters:**

*value* - The string to decode

**Returns:**

a **Long** (p. 2420) object containing the decoded value

**Exceptions:**

**NumberFormatException** if the string is not formatted correctly.

**6.496.2.6 virtual double decaf::lang::Long::doubleValue () const** [inline, virtual]

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.496.2.7 bool decaf::lang::Long::equals (const long long & *l*) const** [inline, virtual]**Parameters:**

*l* - the **Long** (p. 2420) object to compare against.



**Returns:**

true if the two **Integer** (p. 2077) Objects have the same value.

Implements **decaf::lang::Comparable**< **long long** > (p. 1215).

**6.496.2.8 bool decaf::lang::Long::equals (const Long & l) const [inline]****Parameters:**

*l* - the **Long** (p. 2420) object to compare against.

**Returns:**

true if the two **Integer** (p. 2077) Objects have the same value.

**6.496.2.9 virtual float decaf::lang::Long::floatValue () const [inline, virtual]**

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.496.2.10 static long long decaf::lang::Long::highestOneBit (long long value) [static]**

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters:**

*value* - the long long to be inspected

**Returns:**

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.496.2.11 virtual int decaf::lang::Long::intValue () const [inline, virtual]**

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.496.2.12** `virtual long long decaf::lang::Long::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.496.2.13** `static long long decaf::lang::Long::lowestOneBit (long long value) [static]`

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters:**

*value* - the long long to be inspected

**Returns:**

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.496.2.14** `static int decaf::lang::Long::numberOfLeadingZeros (long long value) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values  $x$ :

$\ast \text{floor}(\log_2(x)) = 63 - \text{numberOfLeadingZeros}(x) \ast \text{ceil}(\log_2(x)) = 64 - \text{numberOfLeadingZeros}(x) - 1$

**Parameters:**

*value* - the long long to be inspected

**Returns:**

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

**6.496.2.15** `static int decaf::lang::Long::numberOfTrailingZeros (long long value) [static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

**Parameters:**

*value* - the int to be inspected

**Returns:**

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

**6.496.2.16** `virtual bool decaf::lang::Long::operator< (const long long & l) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*l* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1215).

**6.496.2.17** `virtual bool decaf::lang::Long::operator< (const Long & l) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*l* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.496.2.18** `virtual bool decaf::lang::Long::operator== (const long long & l) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*l* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1215).

**6.496.2.19** `virtual bool decaf::lang::Long::operator== (const Long & l) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*l* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.496.2.20** `static long long decaf::lang::Long::parseLong (const std::string & value,`  
`int radix) throw ( exceptions::NumberFormatException )` [static]

Returns a **Long** (p. 2420) object holding the value extracted from the specified string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2420) object that represents the long long value specified by the string.

**Parameters:**

*value* - **String** (p. 3665) to parse

*radix* - the base encoding of the string

**Returns:**

long long value

**Exceptions:**

*NumberFormatException* on invalid string value

**6.496.2.21** `static long long decaf::lang::Long::parseLong (const std::string & value)`  
`throw ( exceptions::NumberFormatException )` [static]

Parses the string argument as a signed decimal long. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

**Parameters:**

*value* - **String** (p. 3665) to parse

**Returns:**

long long value

**Exceptions:**

*NumberFormatException* on invalid string value

**6.496.2.22 static long long decaf::lang::Long::reverse (long long *value*) [static]**

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

**Parameters:**

*value* - the value whose bits are to be reversed

**Returns:**

the reversed bits long long.

**6.496.2.23 static long long decaf::lang::Long::reverseBytes (long long *value*) [static]**

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

**Parameters:**

*value* - the long long whose bytes we are to reverse

**Returns:**

the reversed long long.

**6.496.2.24 static long long decaf::lang::Long::rotateLeft (long long *value*, int *distance*) [static]**

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

**Parameters:**

*value* - the long long to be inspected

*distance* - the number of bits to rotate

**Returns:**

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

**6.496.2.25 static long long decaf::lang::Long::rotateRight (long long *value*, int *distance*) [static]**

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

**Parameters:**

*value* - the long long to be inspected  
*distance* - the number of bits to rotate

**Returns:**

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

**6.496.2.26** `virtual short decaf::lang::Long::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2837).

**6.496.2.27** `static int decaf::lang::Long::signum (long long value) [static]`

Returns the signum function of the specified value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

**Parameters:**

*value* - the long long to be inspected

**Returns:**

the signum function of the specified long long value.

**6.496.2.28** `static std::string decaf::lang::Long::toBinaryString (long long value) [static]`

Returns a string representation of the long long argument as an unsigned long long in base 2. The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

**Parameters:**

*value* - the long long to be translated to a binary string

**Returns:**

the unsigned long long value as a binary string

**6.496.2.29**    `static std::string decaf::lang::Long::toHexString (long long value)`  
                  [static]

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the `toUpperCase()` method may be called on the result:

**Parameters:**

*value* - the long long to be translated to an Octal string

**Returns:**

the unsigned long long value as a Octal string

**6.496.2.30**    `static std::string decaf::lang::Long::toOctalString (long long value)`  
                  [static]

Returns a string representation of the long long argument as an unsigned long long in base 8. The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

**Parameters:**

*value* - the long long to be translated to an Octal string

**Returns:**

the unsigned long long value as a Octal string

**6.496.2.31**    `static std::string decaf::lang::Long::toString (long long value, int radix)`  
                  [static]**6.496.2.32**    `static std::string decaf::lang::Long::toString (long long value)` [static]

Converts the long to a **String** (p. 3665) representation.

**Parameters:**

*value* The long to convert to a std::string.

**Returns:**

string representation

**6.496.2.33** `std::string decaf::lang::Long::toString () const`**Returns:**

this **Long** (p. 2420) Object as a **String** (p. 3665) Representation

**6.496.2.34** `static Long decaf::lang::Long::valueOf (const std::string & value, int radix) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Long** (p. 2420) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the `parseLong( std::string, int )` method. The result is a **Long** (p. 2420) object that represents the long long value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base ( radix )

*radix* - base of the string to parse.

**Returns:**

new **Long** (p. 2420) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a valid long long.

**6.496.2.35** `static Long decaf::lang::Long::valueOf (const std::string & value) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Long** (p. 2420) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong( std::string )` method. The result is a **Integer** (p. 2077) object that represents the long long value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base 10

**Returns:**

new **Long** (p. 2420) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a decimal long long.

**6.496.2.36** `static Long decaf::lang::Long::valueOf (long long value) [inline, static]`

Returns a **Long** (p. 2420) instance representing the specified `int` value.



**Parameters:**

*value* - the long long to wrap

**Returns:**

the new **Integer** (p. 2077) object wrapping value.

**6.496.3 Field Documentation**

**6.496.3.1** `const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFFLL [static]`

The maximum value that the primitive type can hold.

**6.496.3.2** `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]`

The minimum value that the primitive type can hold.

**6.496.3.3** `const int decaf::lang::Long::SIZE = 64 [static]`

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Long.h`

## 6.497 decaf::internal::nio::LongArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/LongArrayBuffer.h> Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

### Public Member Functions

- **LongArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **IntArrayBuffer** (p. 2056) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **LongArrayBuffer** (long long \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **LongArrayBuffer** (p. 2434) object that wraps the given array.*

- **LongArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **LongArrayBuffer** (const LongArrayBuffer &other)

*Create a **LongArrayBuffer** (p. 2434) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**LongArrayBuffer** ()
- virtual long long \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the long long array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928).*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset long longo the backing array where index zero starts.*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual LongBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only long long buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only long long buffer which the caller then owns.*

- virtual LongBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns:**

*a reference to this **LongBuffer** (p. 2444).*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual LongBuffer \* **duplicate** ()

*Creates a new long long buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

*The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*a new long long **Buffer** (p. 928) which the caller owns.*

- virtual long long **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns:**

*the long long at the current position.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return.

- virtual long long **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the long long is to be read.

**Returns:**

the long long that is located at the given index.

**Exceptions:**

**IndexOutOfBoundsException** if index is not smaller than the buffer's limit, or index is negative.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

- virtual LongBuffer & **put** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters:**

*value* The long longs value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual LongBuffer & **put** (int index, long long value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given long longs long longo this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data  
*value* The long longs to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if index greater than the buffer's limit minus the size of the type being written.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

- virtual LongBuffer \* **slice** () const

Creates a new **LongBuffer** (p. 2444) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **LongBuffer** (p. 2444) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **LongArrayBuffer** (p. 2434) as Read-Only.

### 6.497.1 Constructor & Destructor Documentation

#### 6.497.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (int size, bool readOnly = false) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **IntArrayBuffer** (p. 2056) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

**size** The size of the array, this is the limit we read and write to.

**readOnly** Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

**IllegalArgumentException** if the capacity value is negative.

#### 6.497.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long \* array, int size, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **LongArrayBuffer** (p. 2434) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

**array** The actual array to wrap.

**size** The size of the given array.

**offset** The position that is this buffers start position.

**length** The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

**6.497.1.3** `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer`  
 (const decaf::lang::Pointer< ByteArrayAdapter > &  
*array*, int *offset*, int *length*, bool *readOnly* = false)  
 throw ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset. The capacity and limit of the new **LongArrayBuffer** (p. 2434) will be that of the remaining capacity of the passed buffer.

#### Parameters:

*array* The ByteArrayAdapter to wrap.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if array is NULL

*IndexOutOfBoundsException* if offset + length is greater than array size.

**6.497.1.4** `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer` (const  
 LongArrayBuffer & *other*)

Create a **LongArrayBuffer** (p. 2434) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters:

*other* The **LongArrayBuffer** (p. 2434) this one is to mirror.

**6.497.1.5** `virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer` ()  
 [virtual]

## 6.497.2 Member Function Documentation

**6.497.2.1** `virtual long long* decaf::internal::nio::LongArrayBuffer::array` ()  
 throw ( decaf::lang::exceptions::UnsupportedOperationException,  
 decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2447).

**6.497.2.2** `virtual int decaf::internal::nio::LongArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset long along the backing array where index zero starts.

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2447).

**6.497.2.3** `virtual LongBuffer* de-  
caf::internal::nio::LongArrayBuffer::asReadOnlyBuffer ()  
const [virtual]`

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 2447).

#### 6.497.2.4 **virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ()** **throw ( decaf::nio::ReadOnlyBufferException )** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

##### Returns:

a reference to this **LongBuffer** (p. 2444).

##### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::LongBuffer** (p. 2448).

#### 6.497.2.5 **virtual LongBuffer\* decaf::internal::nio::LongArrayBuffer::duplicate ()** [virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

##### Returns:

a new long long **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2448).

#### 6.497.2.6 **virtual long long decaf::internal::nio::LongArrayBuffer::get (int *index*)** **const throw ( lang::exceptions::IndexOutOfBoundsException )** [virtual]

Absolute get method.

Reads the value at the given index.

##### Parameters:

*index* The index in the **Buffer** (p. 928) where the long long is to be read.

##### Returns:

the long long that is located at the given index.



**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implements **decaf::nio::LongBuffer** (p. 2449).

**6.497.2.7** `virtual long long decaf::internal::nio::LongArrayBuffer::get () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the long long at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implements **decaf::nio::LongBuffer** (p. 2450).

**6.497.2.8** `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::LongBuffer** (p. 2450).

**6.497.2.9** `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 931).

**6.497.2.10** `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (int index, long long value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given long longs long longo this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data

*value* The long longs to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2450).

**6.497.2.11** `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters:**

*value* The long longs value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2451).

**6.497.2.12** `virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 2434) as Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.497.2.13** `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice () const` [virtual]

Creates a new **LongBuffer** (p. 2444) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **LongBuffer** (p. 2444) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2453).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**LongArrayBuffer.h**

## 6.498 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:.

#include <src/main/decaf/nio/LongBuffer.h> Inheritance diagram for decaf::nio::LongBuffer:

### Public Member Functions

- virtual **~LongBuffer** ()
- virtual std::string **toString** () const
- virtual long long \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the long long array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **LongBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only long long buffer that shares this buffer's content.*
- virtual **LongBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **LongBuffer** \* **duplicate** ()=0  
*Creates a new long long buffer that shares this buffer's content.*
- virtual long long **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual long long **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **LongBuffer** & **get** (std::vector< long long > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **LongBuffer** & **get** (long long \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible long long array.*
- **LongBuffer** & **put** (**LongBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException )

*This method transfers the long longs remaining in the given source buffer long longo this buffer.*

- **LongBuffer** & **put** (const long long \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers long longs long longo this buffer from the given source array.*

- **LongBuffer** & **put** (std::vector< long long > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source long longs array long longo this buffer.*

- virtual **LongBuffer** & **put** (long long value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the current position, and then increments the position.*

- virtual **LongBuffer** & **put** (int index, long long value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the given index.*

- virtual **LongBuffer** \* **slice** () const =0

*Creates a new **LongBuffer** (p. 2444) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **LongBuffer** &value) const

- virtual bool **equals** (const **LongBuffer** &value) const

- virtual bool **operator==** (const **LongBuffer** &value) const

- virtual bool **operator<** (const **LongBuffer** &value) const

## Static Public Member Functions

- static **LongBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new Double buffer.*

- static **LongBuffer** \* **wrap** (long long \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **LongBuffer** (p. 2444).*

- static **LongBuffer** \* **wrap** (std::vector< long long > &buffer)

*Wraps the passed STL long long Vector in a **LongBuffer** (p. 2444).*

## Protected Member Functions

- **LongBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **LongBuffer** (p. 2444) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

### 6.498.1 Detailed Description

This class defines four categories of operations upon long long buffers:. o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long longo an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.498.2 Constructor & Destructor Documentation

#### 6.498.2.1 decaf::nio::LongBuffer::LongBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **LongBuffer** (p. 2444) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

#### Parameters:

***capacity*** The size and limit of the **Buffer** (p. 928) in long longs.

#### Exceptions:

***IllegalArgumentException*** if capacity is negative.

#### 6.498.2.2 virtual decaf::nio::LongBuffer::~~LongBuffer () [inline, virtual]

### 6.498.3 Member Function Documentation

#### 6.498.3.1 static LongBuffer\* decaf::nio::LongBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters:**

*capacity* The size of the Double buffer in long longs.

**Returns:**

the **LongBuffer** (p. 2444) that was allocated, caller owns.

**6.498.3.2** `virtual long long* decaf::nio::LongBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the long long array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928).

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2438).

**6.498.3.3** `virtual int decaf::nio::LongBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset long longo the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2439).

**6.498.3.4** `virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new

buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns:

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2439).

#### 6.498.3.5 virtual LongBuffer& decaf::nio::LongBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns:

a reference to this **LongBuffer** (p. 2444).

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2440).

#### 6.498.3.6 virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & value) const [virtual]

#### 6.498.3.7 virtual LongBuffer\* decaf::nio::LongBuffer::duplicate () [pure virtual]

Creates a new long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

a new long long **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2440).



**6.498.3.8** `virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & value) const` [virtual]

**6.498.3.9** `LongBuffer& decaf::nio::LongBuffer::get (long long * buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method. This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies `length` long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

**Parameters:**

*buffer* The pointer to an allocated long long buffer to fill.

*size* The size of the passed in buffer.

*offset* The position in the buffer to start filling.

*length* The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

**BufferUnderflowException** (p. 957) if there are fewer than `length` long longs remaining in this buffer

**NullPolong** longerException if the passed buffer is null.

**IndexOutOfBoundsException** if the preconditions of `size`, `offset`, or `length` are not met.

**6.498.3.10** `LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > buffer) throw ( BufferUnderflowException )`

Relative bulk get method. This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

**BufferUnderflowException** (p. 957) if there are fewer than `length` long longs remaining in this buffer.

**6.498.3.11** `virtual long long decaf::nio::LongBuffer::get (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method. Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the long long is to be read.

**Returns:**

the long long that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or index is negative.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2440).

**6.498.3.12** `virtual long long decaf::nio::LongBuffer::get () throw ( BufferUnderflowException )` [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the long long at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2441).

**6.498.3.13** `virtual bool decaf::nio::LongBuffer::hasArray () const` [pure virtual]

Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2441).

- 6.498.3.14** `virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & value) const` [virtual]
- 6.498.3.15** `virtual bool decaf::nio::LongBuffer::operator== (const LongBuffer & value) const` [virtual]
- 6.498.3.16** `virtual LongBuffer& decaf::nio::LongBuffer::put (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes the given long longs long longo this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data  
*value* The long longs to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2441).

- 6.498.3.17** `virtual LongBuffer& decaf::nio::LongBuffer::put (long long value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters:**

*value* The long longs value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2442).

- 6.498.3.18** `LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & buffer) throw ( BufferOverflowException, ReadOnlyBufferException )`

This method transfers the entire content of the given source long longs array long longo this buffer. This is the same as calling `put( &buffer[0], 0, buffer.size()`.

**Parameters:**

*buffer* The buffer whose contents are copied to this **LongBuffer** (p. 2444).

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**6.498.3.19** **LongBuffer& decaf::nio::LongBuffer::put (const long long \* *buffer*, int *size*, int *offset*, int *length*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers long longs long longo this buffer from the given source array. If there are more long longs to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no long longs are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies length bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

**Parameters:**

*buffer* The array from which long longs are to be read.

*size* The size of the buffer passed.

*offset* The offset within the array of the first char to be read.

*length* The number of long longs to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

**NullPointerException** if the passed buffer is null.

**IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

**6.498.3.20** **LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & *src*) throw ( BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException )**

This method transfers the long longs remaining in the given source buffer long longo this buffer. If there are more long longs remaining in the source buffer than in this buffer, that is, if

`src.remaining() > remaining()` (p. 933), then no long longs are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters:

*src* The buffer to take long longs from an place in this one.

#### Returns:

a reference to this buffer.

#### Exceptions:

**BufferOverflowException** (p. 954) if there is insufficient space in this buffer for the remaining long longs in the source buffer

**IllegalArgumentException** if the source buffer is this buffer

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only

### 6.498.3.21 virtual LongBuffer\* decaf::nio::LongBuffer::slice () const [pure virtual]

Creates a new **LongBuffer** (p. 2444) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

the newly create **LongBuffer** (p. 2444) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2442).

### 6.498.3.22 virtual std::string decaf::nio::LongBuffer::toString () const [virtual]

#### Returns:

a `std::string` describing this object

### 6.498.3.23 static LongBuffer\* decaf::nio::LongBuffer::wrap (std::vector< long long > & buffer) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 2444). The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new **LongBuffer** (p. 2444) that is backed by *buffer*, caller owns.

**6.498.3.24** `static LongBuffer* decaf::nio::LongBuffer::wrap (long  
long * array, int size, int offset, int length)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **LongBuffer** (p. 2444). The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*array* The array that will back the new buffer.

*size* The size of the passed in array.

*offset* The offset of the subarray to be used.

*length* The length of the subarray to be used.

**Returns:**

a new **LongBuffer** (p. 2444) that is backed by *buffer*, caller owns.

**Exceptions:**

*NullPointerException* if the array pointer is NULL.

*IndexOutOfBoundsException* if the preconditions of *size*, *offset*, or *length* are not met.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/LongBuffer.h`

## 6.499 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

### Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

#### 6.499.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different **threads** (p. 96) safely.

#### 6.499.2 Constructor & Destructor Documentation

**6.499.2.1** **activemq::util::LongSequenceGenerator::LongSequenceGenerator** ()

**6.499.2.2** **virtual**  
**activemq::util::LongSequenceGenerator::~~LongSequenceGenerator** ()  
[inline, virtual]

#### 6.499.3 Member Function Documentation

**6.499.3.1** **long long activemq::util::LongSequenceGenerator::getLastSequenceId** ()

**Returns:**

the last id that was generated.

**6.499.3.2** **long long activemq::util::LongSequenceGenerator::getNextSequenceId** ()

**Returns:**

the next id in the sequence.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/LongSequenceGenerator.h`

## 6.500 decaf::net::MalformedURLException Class Reference

#include <src/main/decaf/net/MalformedURLException.h> Inheritance diagram for decaf::net::MalformedURLException:

### Public Member Functions

- **MalformedURLException** () throw ()  
*Default Constructor.*
- **MalformedURLException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()  
*Copy Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **MalformedURLException** (const std::exception \*cause) throw ()  
*Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **MalformedURLException** \* clone () const  
*Clones this exception.*
- virtual ~**MalformedURLException** () throw ()

### 6.500.1 Constructor & Destructor Documentation

#### 6.500.1.1 decaf::net::MalformedURLException::MalformedURLException () throw () [inline]

Default Constructor.

#### 6.500.1.2 decaf::net::MalformedURLException::MalformedURLException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References decaf::lang::Exception::Exception().



**6.500.1.3 decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.500.1.4 decaf::net::MalformedURLException::MalformedURLException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.500.1.5 decaf::net::MalformedURLException::MalformedURLException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.500.1.6 decaf::net::MalformedURLException::MalformedURLException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.500.1.7** `virtual decaf::net::MalformedURLException::~~MalformedURLException  
() throw () [inline, virtual]`

## **6.500.2 Member Function Documentation**

**6.500.2.1** `virtual MalformedURLException* de-  
caf::net::MalformedURLException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/MalformedURLException.h`

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

#include <src/main/decaf/util/Map.h> Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

### Data Structures

- class **Entry**

### Public Member Functions

- **Map** ()  
*Default constructor - does nothing.*
- virtual ~**Map** ()
- virtual bool **equals** (const **Map** &source) const =0  
*Comparison, equality is dependent on the method of determining if the element are equal.*
- virtual void **copy** (const **Map** &source)=0  
*Copies the content of the source map into this map.*
- virtual void **clear** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*
- virtual bool **containsKey** (const K &key) const =0  
*Indicates whether or this map contains a value for the given key.*
- virtual bool **containsValue** (const V &value) const =0  
*Indicates whether or this map contains a value for the given value, i.e.*
- virtual bool **isEmpty** () const =0
- virtual std::size\_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2459).*
- virtual const V & **get** (const K &key) const =0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2459).*
- virtual void **put** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Sets the value for the specified key.*

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.*
- virtual V **remove** (const K &key)=0 throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*
- virtual std::vector< K > **keySet** () const =0  
*Returns a **Set** (p. 3439) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

### 6.501.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >
```

**Map** (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

### 6.501.2 Constructor & Destructor Documentation

**6.501.2.1** template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map () [inline]

Default constructor - does nothing.

**6.501.2.2** template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~~Map () [inline, virtual]

### 6.501.3 Member Function Documentation

**6.501.3.1** template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear () throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]

Removes all keys and values from this map.

#### Exceptions:

*UnsupportedOperationException* if this map is unmodifiable.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1237), decaf::util::StlMap< K, V, COMPARATOR > (p. 3606), decaf::util::concurrent::ConcurrentStlMap< Pointer<

MessageId >, Pointer< Message >, MessageId::COMPARATOR >  
 (p. 1237), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-  
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR  
 > (p. 1237), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-  
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR  
 > (p. 1237), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId  
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1237),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,  
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1237),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<  
 ProducerState >, ProducerId::COMPARATOR > (p. 1237), decaf::util::StlMap<  
 cms::Session \*, SessionResolver \* > (p. 3606), decaf::util::StlMap< std::string,  
 WireFormatFactory \* > (p. 3606), decaf::util::StlMap< std::string, PrimitiveVal-  
 ueNode > (p. 3606), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3606),  
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, com-  
 mands::ProducerId::COMPARATOR > (p. 3606), decaf::util::StlMap< std::string,  
 CachedConsumer \* > (p. 3606), decaf::util::StlMap< Pointer< commands::ConsumerId  
 >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3606),  
 decaf::util::StlMap< std::string, TransportFactory \* > (p. 3606), decaf::util::StlMap<  
 Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR  
 > (p. 3606), decaf::util::StlMap< int, Pointer< Command > > (p. 3606),  
 decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, com-  
 mands::ConsumerId::COMPARATOR > (p. 3606), decaf::util::StlMap< std::string,  
 CachedProducer \* > (p. 3606), and decaf::util::StlMap< std::string, cms::Topic \* >  
 (p. 3606).

**6.501.3.2** template<typename K, typename V, typename COMPARATOR =  
 std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR  
 >::containsKey (const K & *key*) const [pure virtual]

Indicates whether or this map contains a value for the given key.

#### Parameters:

*key* The key to look up.

#### Returns:

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COM-  
 PARATOR > (p. 1238), decaf::util::StlMap< K, V, COMPARA-  
 TOR > (p. 3606), decaf::util::concurrent::ConcurrentStlMap< Pointer<  
 MessageId >, Pointer< Message >, MessageId::COMPARATOR >  
 (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-  
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR  
 > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-  
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR  
 > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId  
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1238),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,  
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1238),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<  
 ProducerState >, ProducerId::COMPARATOR > (p. 1238), decaf::util::StlMap<

cms::Session \*, SessionResolver \* > (p. 3606), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3606), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3606), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3606), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3606), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3606), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3606), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3606), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3606), decaf::util::StlMap< int, Pointer< Command > > (p. 3606), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3606), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3606), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3606).

**6.501.3.3** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue (const V & value) const` [pure virtual]

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters:

*value* The Value to look up.

#### Returns:

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1238), decaf::util::StlMap< K, V, COMPARATOR > (p. 3607), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1238), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1238), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3607), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3607), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3607), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3607), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3607), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3607), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3607), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3607), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3607), decaf::util::StlMap< int, Pointer< Command > > (p. 3607),

decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3607), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3607), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3607).

**6.501.3.4** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source) [pure virtual]`

Copies the content of the source map into this map. Erases all existing data in this map.

**Parameters:**

*source* The source object to copy from.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1239), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3607).

**6.501.3.5** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [pure virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

**Parameters:**

*source* - `Map` (p. 2459) to compare to this one.

**Returns:**

true if the `Map` (p. 2459) passed is equal in value to this one.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1239), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3608).

**6.501.3.6** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::get (const K & key) const throw ( lang::exceptions::NoSuchElementException ) [pure virtual]`

Gets the value mapped to the specified key in the `Map` (p. 2459). If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

**Parameters:**

*key* The search key.

**Returns:**

A {const} reference to the value for the given key.

**Exceptions:**

*NoSuchElementException* if the key requests doesn't exist in the **Map** (p. 2459).

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 1239), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 3608), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 1239), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 1239), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 1239), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 1239), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId** >, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 1239), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 1239), **decaf::util::StlMap**< **cms::Session** \*, **SessionResolver** \* > (p. 3608), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** \* > (p. 3608), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 3608), **decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3608), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR** > (p. 3608), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3608), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*, **commands::ConsumerId::COMPARATOR** > (p. 3608), **decaf::util::StlMap**< **std::string**, **TransportFactory** \* > (p. 3608), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3608), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3608), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3608), **decaf::util::StlMap**< **std::string**, **CachedProducer** \* > (p. 3608), and **decaf::util::StlMap**< **std::string**, **cms::Topic** \* > (p. 3608).

**6.501.3.7** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get (const K & key) throw ( lang::exceptions::NoSuchElementException ) [pure virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2459). If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

**Parameters:**

*key* The search key.

**Returns:**

A reference to the value for the given key.

**Exceptions:**

*NoSuchElementException* if the key requests doesn't exist in the **Map** (p. 2459).

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 1240), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 3608), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**<



MessageId >, Pointer< Message >, MessageId::COMPARATOR >  
 (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-  
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR  
 > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-  
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR  
 > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId  
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1240),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,  
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1240),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<  
 ProducerState >, ProducerId::COMPARATOR > (p. 1240), decaf::util::StlMap<  
 cms::Session \*, SessionResolver \* > (p. 3608), decaf::util::StlMap< std::string,  
 WireFormatFactory \* > (p. 3608), decaf::util::StlMap< std::string, PrimitiveVal-  
 ueNode > (p. 3608), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3608),  
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, com-  
 mands::ProducerId::COMPARATOR > (p. 3608), decaf::util::StlMap< std::string,  
 CachedConsumer \* > (p. 3608), decaf::util::StlMap< Pointer< commands::ConsumerId  
 >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3608),  
 decaf::util::StlMap< std::string, TransportFactory \* > (p. 3608), decaf::util::StlMap<  
 Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR  
 > (p. 3608), decaf::util::StlMap< int, Pointer< Command > > (p. 3608),  
 decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, com-  
 mands::ConsumerId::COMPARATOR > (p. 3608), decaf::util::StlMap< std::string,  
 CachedProducer \* > (p. 3608), and decaf::util::StlMap< std::string, cms::Topic \* >  
 (p. 3608).

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::equals(), and  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >,  
 ProducerId::COMPARATOR >::equals().

**6.501.3.8** `template<typename K, typename V, typename COMPARATOR =  
 std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR  
 >::isEmpty () const [pure virtual]`

#### Returns:

if the **Map** (p. 2459) contains any element or not, TRUE or FALSE

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COM-  
 PARATOR > (p. 1240), decaf::util::StlMap< K, V, COMPARA-  
 TOR > (p. 3609), decaf::util::concurrent::ConcurrentStlMap< Pointer<  
 MessageId >, Pointer< Message >, MessageId::COMPARATOR >  
 (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-  
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR  
 > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-  
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR  
 > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId  
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1240),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,  
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1240),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<  
 ProducerState >, ProducerId::COMPARATOR > (p. 1240), decaf::util::StlMap<  
 cms::Session \*, SessionResolver \* > (p. 3609), decaf::util::StlMap< std::string,  
 WireFormatFactory \* > (p. 3609), decaf::util::StlMap< std::string, PrimitiveVal-

ueNode > (p. 3609), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3609), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< int, Pointer< Command > > (p. 3609), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3609), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3609).

**6.501.3.9** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> virtual std::vector<K> decaf::util::Map< K, V,  
COMPARATOR >::keySet () const [pure virtual]`

Returns a **Set** (p. 3439) view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2155), **Set.remove** (p. 187), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

#### Returns:

the entire set of keys in this map as a std::vector.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1240), decaf::util::StlMap< K, V, COMPARATOR > (p. 3609), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1240), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1240), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3609), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3609), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3609), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3609), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< int, Pointer< Command > > (p. 3609),

decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3609), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3609).

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

```
6.501.3.10  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::Map< K, V,
            COMPARATOR >::put (const K & key, const V & value) throw (
            decaf::lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Sets the value for the specified key.

#### Parameters:

*key* The target key.

*value* The value to be set.

#### Exceptions:

*UnsupportedOperationException* if this map is unmodifiable.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1242), decaf::util::StlMap< K, V, COMPARATOR > (p. 3610), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1242), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1242), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1242), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1242), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1242), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1242), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3610), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3610), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3610), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3610), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3610), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3610), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< int, Pointer< Command > > (p. 3610), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3610), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3610).

**6.501.3.11** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.

**Parameters:**

*other* A **Map** (p. 2459) instance whose elements are to all be inserted in this **Map** (p. 2459).

**Exceptions:**

**UnsupportedOperationException** If the implementing class does not support the putAll operation.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1242), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3611), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1242), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1242), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1242), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1242), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1242), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1242), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3611), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3611), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3611), and `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3611).

**6.501.3.12** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::Map< K, V, COMPARATOR >::remove (const K & key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

**Parameters:**

*key* The search key.

**Returns:**

a copy of the element that was previously mapped to the given key

**Exceptions:**

*NoSuchElementException* if this key is not in the **Map** (p. 2459).

*UnsupportedOperationException* if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1244), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3611), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1244), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1244), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1244), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1244), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1244), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1244), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3611), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3611), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3611), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3611), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3611), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3611), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3611), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3611), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3611), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3611), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3611), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3611), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3611).

**6.501.3.13** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> virtual std::size_t decaf::util::Map< K, V,  
COMPARATOR >::size () const [pure virtual]`

**Returns:**

The number of elements (key/value pairs) in this map.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1246), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3612), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1246), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1246), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1246), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1246), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1246),

decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1246), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3612), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3612), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3612), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3612), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3612), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3612), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< int, Pointer< Command > > (p. 3612), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3612), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3612).

**6.501.3.14** `template<typename K, typename V, typename COMPARATOR  
= std::less<K>> virtual std::vector<V> decaf::util::Map< K, V,  
COMPARATOR >::values () const [pure virtual]`

#### Returns:

the entire set of values in this map as a std::vector.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1246), decaf::util::StlMap< K, V, COMPARATOR > (p. 3612), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1246), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1246), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1246), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1246), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1246), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1246), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3612), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3612), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3612), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3612), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3612), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3612), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< int, Pointer< Command > > (p. 3612), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3612), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3612), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3612).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

## 6.502 cms::MapMessage Class Reference

A **MapMessage** (p. 2472) object is used to send a set of name-value pairs.

#include <src/main/cms/MapMessage.h> Inheritance diagram for cms::MapMessage:

### Public Member Functions

- virtual **~MapMessage** ()
- virtual std::vector< std::string > **getMapNames** () const =0 throw ( CMSEException )  
*Returns an Enumeration of all the names in the **MapMessage** (p. 2472) object.*
- virtual bool **itemExists** (const std::string &name) const =0 throw ( CMSEException )  
*Indicates whether an item exists in this **MapMessage** (p. 2472) object.*
- virtual bool **getBoolean** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const std::string &name, bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a boolean value with the specified name into the Map.*
- virtual unsigned char **getByte** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Char value of the Specified name.*
- virtual void **setChar** (const std::string &name, char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Char value with the specified name into the Map.*
- virtual double **getDouble** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )



*Returns the Double value of the Specified name.*

- virtual void **setDouble** (const std::string &name, double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Double value with the specified name into the Map.*

- virtual float **getFloat** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Float value of the Specified name.*

- virtual void **setFloat** (const std::string &name, float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Float value with the specified name into the Map.*

- virtual int **getInt** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Int value of the Specified name.*

- virtual void **setInt** (const std::string &name, int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Int value with the specified name into the Map.*

- virtual long long **getLong** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Long value of the Specified name.*

- virtual void **setLong** (const std::string &name, long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Long value with the specified name into the Map.*

- virtual short **getShort** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Short value of the Specified name.*

- virtual void **setShort** (const std::string &name, short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Short value with the specified name into the Map.*

- virtual std::string **getString** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the String value of the Specified name.*

- virtual void **setString** (const std::string &name, const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a String value with the specified name into the Map.*

### 6.502.1 Detailed Description

A **MapMessage** (p.2472) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The

names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 2472) inherits from the **Message** (p. 2534) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 2472), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 2724) is thrown. To place the **MapMessage** (p. 2472) back into a state where it can be read from and written to, call the `clearBody` method.

**MapMessage** (p. 2472) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1160). The String-to-primitive conversions may throw a **MessageFormatException** (p. 2662) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X								X	
byte		X	X		X	X			X	
short			X		X	X			X	
char				X					X	
int					X	X			X	
long						X			X	
float							X	X	X	
double								X	X	
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.0

## 6.502.2 Constructor & Destructor Documentation

**6.502.2.1** `virtual cms::MapMessage::~~MapMessage () [inline, virtual]`

## 6.502.3 Member Function Documentation

**6.502.3.1** `virtual bool cms::MapMessage::getBoolean (const std::string & name)  
const throw ( cms::MessageFormatException, cms::CMSEException )  
[pure virtual]`

Returns the Boolean value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

**CMSEException** (p. 1160) - if the operation fails due to an internal error.

**MessageFormatException** - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 364).

**6.502.3.2** `virtual unsigned char cms::MapMessage::getByte (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Byte value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 365).

**6.502.3.3** `virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Bytes value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 365).

**6.502.3.4** `virtual char cms::MapMessage::getChar (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Char value of the Specified name.

**Parameters:**

*name* name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 365).

**6.502.3.5** virtual double cms::MapMessage::getDouble (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the Double value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 366).

**6.502.3.6** virtual float cms::MapMessage::getFloat (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the Float value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 366).

**6.502.3.7** virtual int cms::MapMessage::getInt (const std::string & *name*) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the Int value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 366).

**6.502.3.8**    `virtual long long cms::MapMessage::getLong (const std::string & name)  
const throw ( cms::MessageFormatException, cms::CMSException )  
[pure virtual]`

Returns the Long value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 367).

**6.502.3.9**    `virtual std::vector< std::string > cms::MapMessage::getMapNames ()  
const throw ( CMSException ) [pure virtual]`

Returns an Enumeration of all the names in the `MapMessage` (p. 2472) object.

**Returns:**

STL Vector of String values, each of which is the name of an item in the `MapMessage` (p. 2472)

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 367).

**6.502.3.10**    `virtual short cms::MapMessage::getShort (const std::string & name)  
const throw ( cms::MessageFormatException, cms::CMSException )  
[pure virtual]`

Returns the Short value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 368).

**6.502.3.11** `virtual std::string cms::MapMessage::getString (const std::string & name) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the String value of the Specified name.

**Parameters:**

*name* Name of the value to fetch from the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 368).

**6.502.3.12** `virtual bool cms::MapMessage::itemExists (const std::string & name) const throw ( CMSException ) [pure virtual]`

Indicates whether an item exists in this `MapMessage` (p. 2472) object.

**Parameters:**

*name* String name of the Object in question

**Returns:**

boolean value indicating if the name is in the map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 368).

**6.502.3.13** `virtual void cms::MapMessage::setBoolean (const std::string & name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Sets a boolean value with the specified name into the Map.

**Parameters:**

*name* the name of the boolean

*value* the boolean value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWritableException* - if the `Message` (p. 2534) is in Read-only Mode.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 369).

**6.502.3.14** `virtual void cms::MapMessage::setByte (const std::string & name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Byte value with the specified name into the Map.

**Parameters:**

*name* the name of the Byte

*value* the Byte value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 369).

**6.502.3.15** `virtual void cms::MapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Bytes value with the specified name into the Map.

**Parameters:**

*name* The name of the Bytes

*value* The Bytes value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 370).

**6.502.3.16** `virtual void cms::MapMessage::setChar (const std::string & name, char value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Char value with the specified name into the Map.

**Parameters:**

*name* the name of the Char

*value* the Char value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 370).

**6.502.3.17** `virtual void cms::MapMessage::setDouble (const std::string & name, double value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Double value with the specified name into the Map.

**Parameters:**

*name* The name of the Double

*value* The Double value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 370).

**6.502.3.18** `virtual void cms::MapMessage::setFloat (const std::string & name, float value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Float value with the specified name into the Map.

**Parameters:**

*name* The name of the Float

*value* The Float value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 371).

**6.502.3.19** `virtual void cms::MapMessage::setInt (const std::string & name, int value) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets a Int value with the specified name into the Map.

**Parameters:**

*name* The name of the Int



*value* The Int value to set in the Map

**Exceptions:**

*CMSEException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 371).

**6.502.3.20** `virtual void cms::MapMessage::setLong (const std::string & name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets a Long value with the specified name into the Map.

**Parameters:**

*name* The name of the Long

*value* The Long value to set in the Map

**Exceptions:**

*CMSEException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 371).

**6.502.3.21** `virtual void cms::MapMessage::setShort (const std::string & name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets a Short value with the specified name into the Map.

**Parameters:**

*name* The name of the Short

*value* The Short value to set in the Map

**Exceptions:**

*CMSEException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 372).

**6.502.3.22** `virtual void cms::MapMessage::setString (const std::string & name,  
const std::string & value) throw ( cms::MessageNotWriteableException,  
cms::CMSException )` [pure virtual]

Sets a String value with the specified name into the Map.

**Parameters:**

*name* The name of the String

*value* The String value to set in the Map

**Exceptions:**

*CMSException* (p. 1160) - if the operation fails due to an internal error.

*MessageNotWriteableException* (p. 2724) - if the **Message** (p. 2534) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 372).

The documentation for this class was generated from the following file:

- `src/main/cms/MapMessage.h`

## 6.503 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

### Public Member Functions

- **MarkBlockLogger** (**Logger** \*logger, const std::string &blockName)  
*Constructor - Marks Block entry.*
- virtual ~**MarkBlockLogger** ()

### 6.503.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

### 6.503.2 Constructor & Destructor Documentation

#### 6.503.2.1 decaf::util::logging::MarkBlockLogger::MarkBlockLogger (**Logger** \*logger, const std::string & blockName) [inline]

Constructor - Marks Block entry.

#### Parameters:

*logger* **Logger** (p.2386) to use  
*blockName* Block name

#### 6.503.2.2 virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/MarkBlockLogger.h

## 6.504 activemq::wireformat::MarshalAware Class Reference

#include <src/main/activemq/wireformat/MarshalAware.h> Inheritance diagram for activemq::wireformat::MarshalAware:

### Public Member Functions

- virtual `~MarshalAware ()`
- virtual `bool isMarshalAware () const =0`  
*Determine if the class implementing this interface is really wanting to be told about marshaling.*
- virtual `void beforeMarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called before marshaling is started to prepare the object to be marshaled.*
- virtual `void afterMarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called after marshaling is started to cleanup the object being marshaled.*
- virtual `void beforeUnmarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called before unmarshaling is started to prepare the object to be unmarshaled.*
- virtual `void afterUnmarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*
- virtual `void setMarshaledForm (WireFormat *wireFormat, const std::vector< char > &data)=0`  
*Called to set the data to this object that will contain the objects marshaled form.*
- virtual `std::vector< unsigned char > getMarshaledForm (WireFormat *wireFormat)=0`  
*Called to get the data to this object that will contain the objects marshaled form.*

### 6.504.1 Constructor & Destructor Documentation

- 6.504.1.1** `virtual activemq::wireformat::MarshalAware::~~MarshalAware ()`  
[inline, virtual]

### 6.504.2 Member Function Documentation

- 6.504.2.1** `virtual void activemq::wireformat::MarshalAware::afterMarshal (WireFormat * wireFormat) throw ( decaf::io::IOException )` [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

**Parameters:**

*wireFormat* - the **wireformat** (p. 105) object to control marshaling

**6.504.2.2** virtual void activemq::wireformat::MarshalAware::afterUnmarshal  
(WireFormat \* *wireFormat*) throw ( decaf::io::IOException ) [pure  
virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

**Parameters:**

*wireFormat* - the **wireformat** (p. 105) object to control unmarshaling

**6.504.2.3** virtual void activemq::wireformat::MarshalAware::beforeMarshal  
(WireFormat \* *wireFormat*) throw ( decaf::io::IOException ) [pure  
virtual]

Called before marshaling is started to prepare the object to be marshaled.

**Parameters:**

*wireFormat* - the **wireformat** (p. 105) object to control marshaling

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 363), and **activemq::commands::ActiveMQTextMessage** (p. 663).

**6.504.2.4** virtual void activemq::wireformat::MarshalAware::beforeUnmarshal  
(WireFormat \* *wireFormat*) throw ( decaf::io::IOException ) [pure  
virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

**Parameters:**

*wireFormat* - the **wireformat** (p. 105) object to control unmarshaling

**6.504.2.5** virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm  
(WireFormat \* *wireFormat*) [pure virtual]

Called to get the data to this object that will contain the objects marshaled form.

**Parameters:**

*wireFormat* - the **wireformat** (p. 105) object to control unmarshaling

**Returns:**

buffer that holds the objects data.

### 6.504.2.6 `virtual bool activemq::wireformat::MarshalAware::isMarshalAware ()` `const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling. Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

#### Returns:

true if this class cares about marshaling.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 368), `activemq::commands::BaseDataStructure` (p. 832), `activemq::commands::Message` (p. 2526), and `activemq::commands::WireFormatInfo` (p. 3987).

### 6.504.2.7 `virtual void activemq::wireformat::MarshalAware::setMarshaledForm` `(WireFormat * wireFormat, const std::vector< char > & data) [pure virtual]`

Called to set the data to this object that will contain the objects marshaled form.

#### Parameters:

*wireFormat* - the `wireformat` (p. 105) object to control unmarshaling  
*data* - vector of object binary data

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/MarshalAware.h`

## 6.505 activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure (OpenWireFormat *format)`

#### 6.505.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.505.2 Constructor & Destructor Documentation

- 6.505.2.1 virtual  
activemq::wireformat::openwire::marshal::v6::MarshallerFactory::~~MarshallerFactory  
( ) [inline, virtual]

#### 6.505.3 Member Function Documentation

- 6.505.3.1 virtual void ac-  
tivemq::wireformat::openwire::marshal::v6::MarshallerFactory::configure  
(OpenWireFormat \* *format*) [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h`

## 6.506 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

#### 6.506.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.506.2 Constructor & Destructor Documentation

- 6.506.2.1** virtual  
`activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

#### 6.506.3 Member Function Documentation

- 6.506.3.1** virtual void `activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h`



## 6.507 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure (OpenWireFormat *format)`

#### 6.507.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.507.2 Constructor & Destructor Documentation

**6.507.2.1** virtual  
`activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

#### 6.507.3 Member Function Documentation

**6.507.3.1** virtual void `activemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure (OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h`

## 6.508 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

#### 6.508.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.508.2 Constructor & Destructor Documentation

- 6.508.2.1** virtual  
`activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

#### 6.508.3 Member Function Documentation

- 6.508.3.1** virtual void `activemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h`

## 6.509 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure (OpenWireFormat *format)`

#### 6.509.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.509.2 Constructor & Destructor Documentation

- 6.509.2.1 virtual  
activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~MarshallerFactory  
( ) [inline, virtual]

#### 6.509.3 Member Function Documentation

- 6.509.3.1 virtual void ac-  
tivemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure  
(OpenWireFormat \* *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h**

## 6.510 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

#### 6.510.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.510.2 Constructor & Destructor Documentation

- 6.510.2.1** virtual  
`activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

#### 6.510.3 Member Function Documentation

- 6.510.3.1** virtual void `activemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h`

## 6.511 activemq::util::MarshallingSupport Class Reference

```
#include <src/main/activemq/util/MarshallingSupport.h>
```

### Public Member Functions

- **MarshallingSupport** ()
- virtual **~MarshallingSupport** ()

### Static Public Member Functions

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )

*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*

- static void **writeString16** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )

*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*

- static void **writeString32** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )

*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*

- static std::string **readString16** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )

*Reads an Openwire encoded string from the provided DataInputStream.*

- static std::string **readString32** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )

*Reads an Openwire encoded string from the provided DataInputStream.*

- static std::string **asciiToModifiedUtf8** (const std::string &asciiString) throw ( decaf::io::UTFDataFormatException )

*Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.*

- static std::string **modifiedUtf8ToAscii** (const std::string modifiedUtf8String) throw ( decaf::io::UTFDataFormatException )

*Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].*

## 6.511.1 Constructor & Destructor Documentation

6.511.1.1 `activemq::util::MarshallingSupport::MarshallingSupport ()`

6.511.1.2 `virtual activemq::util::MarshallingSupport::~~MarshallingSupport ()`  
[virtual]

## 6.511.2 Member Function Documentation

6.511.2.1 `static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8 (const std::string & asciiString) throw ( decaf::io::UTFDataFormatException )` [static]

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string. This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

### Parameters:

*asciiString* The ASCII string to encode as Modified UTF-8

### Returns:

a string containing the Modified UTF-8 encoded form of the provided string.

### Exceptions:

*UTFDataFormatException* if the length of the encoded string would exceed the size of an signed integer.

6.511.2.2 `static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii (const std::string modifiedUtf8String) throw ( decaf::io::UTFDataFormatException )` [static]

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255]. This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

### Parameters:

*modifiedUtf8String* The string to convert from Modified UTF-8 to ASCII.

### Returns:

the ASCII encoded version of the provided string.

### Exceptions:

*UTFDataFormatException* if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.

**6.511.2.3**    `static std::string activemq::util::MarshallingSupport::readString16  
(decaf::io::DataInputStream & dataIn) throw ( decaf::io::IOException )  
[static]`

Reads an Openwire encoded string from the provided DataInputStream. No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

**Parameters:**

*dataIn* The DataInputStream to read the String data from.

**Returns:**

the String value.

**Exceptions:**

*IOException* if an I/O error occurs while writing the string.

**6.511.2.4**    `static std::string activemq::util::MarshallingSupport::readString32  
(decaf::io::DataInputStream & dataIn) throw ( decaf::io::IOException )  
[static]`

Reads an Openwire encoded string from the provided DataInputStream. No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

**Parameters:**

*dataIn* The DataInputStream to read the String data from.

**Returns:**

the String value.

**Exceptions:**

*IOException* if an I/O error occurs while writing the string.

**6.511.2.5**    `static void activemq::util::MarshallingSupport::writeString  
(decaf::io::DataOutputStream & dataOut, const std::string & value)  
throw ( decaf::io::IOException ) [static]`

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string. User must encode to Modified UTF-8 as needed.

**Parameters:**

*dataOut* The DataOutputStream to write the String data to.

*value* Thre String value to write in Openwire form.

**Exceptions:**

*IOException* if an I/O error occurs while writing the string.

**6.511.2.6** static void activemq::util::MarshallingSupport::writeString16  
(decaf::io::DataOutputStream & *dataOut*, const std::string & *value*)  
throw ( decaf::io::IOException ) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string. User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

**Parameters:**

*dataOut* The DataOutputStream to write the String data to.

*value* Thre String value to write in Openwire form.

**Exceptions:**

*IOException* if an I/O error occurs while writing the string.

**6.511.2.7** static void activemq::util::MarshallingSupport::writeString32  
(decaf::io::DataOutputStream & *dataOut*, const std::string & *value*)  
throw ( decaf::io::IOException ) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string. User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

**Parameters:**

*dataOut* The DataOutputStream to write the String data to.

*value* Thre String value to write in Openwire form.

**Exceptions:**

*IOException* if an I/O error occurs while writing the string.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MarshallingSupport.h**



## 6.512 decaf::lang::Math Class Reference

The class `Math` (p. 2497) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

```
#include <src/main/decaf/lang/Math.h>
```

### Public Member Functions

- `Math ()`
- `virtual ~Math ()`

### Static Public Member Functions

- static int **abs** (int value)  
*Returns the absolute value of an int value.*
- static long long **abs** (long long value)  
*Returns the absolute value of an long long value.*
- static float **abs** (float value)  
*Returns the absolute value of a float value.*
- static double **abs** (double value)  
*Returns the absolute value of a double value.*
- static double **sqrt** (double value)  
*Returns the arc cosine of an angle, in the range of 0.0 through pi.*
- static double **pow** (double base, double exp)  
*Returns the value of the first argument raised to the power of the second argument.*
- static short **min** (short a, short b)  
*Returns the double value that is closest in value to the argument and is equal to a mathematical integer.*
- static int **min** (int a, int b)  
*Returns the smaller of two int values.*
- static unsigned int **min** (unsigned int a, unsigned int b)  
*Returns the smaller of two unsigned int values.*
- static long long **min** (long long a, long long b)  
*Returns the smaller of two long long values.*
- static float **min** (float a, float b)  
*Returns the smaller of two float values.*
- static double **min** (double a, double b)  
*Returns the smaller of two double values.*

- static short **max** (short a, short b)  
*Returns the larger of two **short** values.*
- static int **max** (int a, int b)  
*Returns the larger of two **int** values.*
- static long long **max** (long long a, long long b)  
*Returns the larger of two **long long** values.*
- static float **max** (float a, float b)  
*Returns the greater of two float values.*
- static double **max** (double a, double b)  
*Returns the greater of two double values.*
- static double **ceil** (double value)  
*Returns the natural logarithm (base e) of a double value.*
- static double **floor** (double value)  
*Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.*
- static int **round** (float value)  
*Returns the closest int to the argument.*
- static long long **round** (double value)  
*Returns the closest long long to the argument.*
- static double **random** ()  
*Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.*
- static float **signum** (float value)  
*Returns Euler's number e raised to the power of a double value.*
- static double **signum** (double value)  
*Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.*
- static double **toRadians** (double angdeg)  
*Returns the measure in radians of the supplied degree angle.*
- static double **toDegrees** (double angrad)  
*Returns the measure in degrees of the supplied radian angle.*

## Static Public Attributes

- static const double **E**
- static const double **PI**

## 6.512.1 Detailed Description

The class `Math` (p. 2497) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

## 6.512.2 Constructor & Destructor Documentation

**6.512.2.1** `decaf::lang::Math::Math ()` [inline]

**6.512.2.2** `virtual decaf::lang::Math::~~Math ()` [inline, virtual]

## 6.512.3 Member Function Documentation

**6.512.3.1** `static double decaf::lang::Math::abs (double value)` [static]

Returns the absolute value of a double value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Double::longBitsToDouble` (p. 1794)( `0x7fffffffffffffffULL & Double::doubleToLongBits( value )` )

### Parameters:

*value* - the value to return the abs of

### Returns:

the value if positive, otherwise the negative of value

**6.512.3.2** `static float decaf::lang::Math::abs (float value)` [static]

Returns the absolute value of a float value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Float::intBitsToFloat` (p. 1909)( `0x7fffffff & Float::floatToIntBits( value )` )

### Parameters:

*value* - the value to return the abs of

### Returns:

the value if positive, otherwise the negative of value

**6.512.3.3** `static long long decaf::lang::Math::abs (long long value)` [inline, static]

Returns the absolute value of an long long value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

**Parameters:**

*value* - the value to return the abs of

**Returns:**

the value if positive, otherwise the negative of value

**6.512.3.4 static int decaf::lang::Math::abs (int *value*) [inline, static]**

Returns the absolute value of an int value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

**Parameters:**

*value* - the value to return the abs of

**Returns:**

the value if positive, otherwise the negative of value

**6.512.3.5 static double decaf::lang::Math::ceil (double *value*) [static]**

Returns the natural logarithm (base e) of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

**Parameters:**

*value* the value to compute the natural log of.

**Returns:**

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to  $10^n$  for integer  $n$ , then the result is  $n$ .

**Parameters:**

*value* - the value to operate on

**Returns:**

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values  $x$ , the result of  $\log_{10}(x)$  is much closer to the true result of  $\ln(1 + x)$  than the floating-point evaluation of  $\log(1.0+x)$ .

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the value to operate on

**Returns:**

the the value  $\ln(x + 1)$ , the natural log of  $x + 1$  Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of `Math.ceil(x)` is exactly the value of `-Math.floor(-x)`.

**Parameters:**

*value* - the value to find the ceiling of

**Returns:**

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

**6.512.3.6 static double decaf::lang::Math::floor (double *value*) [static]**

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

**Parameters:**

*value* - the value to find the floor of

**Returns:**

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

**6.512.3.7 static double decaf::lang::Math::max (double *a*, double *b*) [static]**

Returns the greater of two double values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

**Parameters:**

*a* - an argument.

***b*** - another argument.

**Returns:**

the larger of  $a$  and  $b$ .

### 6.512.3.8 static float decaf::lang::Math::max (float *a*, float *b*) [static]

Returns the greater of two float values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

### Parameters:

*a* - an argument.

***b*** - another argument.

**Returns:**

the larger of **a** and **b**.

```
6.512.3.9 static long long decaf::lang::Math::max (long long a, long long b)
[inline, static]
```

Returns the larger of two `long long` values. That is, the result the argument closer to the value of `Long::MAX_VALUE` (p.2433). If the arguments have the same value, the result is that same value.

### Parameters:

*a* - an argument.

*b* - another argument.

**Returns:**

the larger of  $a$  and  $b$ .

```
6.512.3.10 static int decaf::lang::Math::max (int a, int b) [inline, static]
```

Returns the larger of two `int` values. That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p.2090). If the arguments have the same value, the result is that same value.

### Parameters:

*a* - an argument.

***b*** - another argument.

**Returns:**

the larger of **a** and **b**.

**6.512.3.11 static short decaf::lang::Math::max (short *a*, short *b*) [inline, static]**

Returns the larger of two `short` values. That is, the result the argument closer to the value of `Short::MAX_VALUE` (p.3448). If the arguments have the same value, the result is that same value.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the larger of *a* and *b*.

**6.512.3.12 static double decaf::lang::Math::min (double *a*, double *b*) [static]**

Returns the smaller of two double values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.13 static float decaf::lang::Math::min (float *a*, float *b*) [static]**

Returns the smaller of two float values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.14** `static long long decaf::lang::Math::min (long long a, long long b)`  
[inline, static]

Returns the smaller of two `long long` values. That is, the result the argument closer to the value of `Long::MIN_VALUE` (p. 2433). If the arguments have the same value, the result is that same value.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.15** `static unsigned int decaf::lang::Math::min (unsigned int a, unsigned int b)`  
[inline, static]

Returns the smaller of two `unsigned int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 2090). If the arguments have the same value, the result is that same value.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.16** `static int decaf::lang::Math::min (int a, int b)` [inline, static]

Returns the smaller of two `int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 2090). If the arguments have the same value, the result is that same value.

**Parameters:**

- a* - an argument.
- b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.17** `static short decaf::lang::Math::min (short a, short b)` [inline, static]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:



o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

**Parameters:**

*value* - the value to round to the nearest integer

**Returns:**

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf.lang.Short::MIN_VALUE` (p. 3448). If the arguments have the same value, the result is that same value.

**Parameters:**

*a* - an argument.

*b* - another argument.

**Returns:**

the smaller of *a* and *b*.

**6.512.3.18 static double decaf::lang::Math::pow (double *base*, double *exp*)**  
[static]

Returns the value of the first argument raised to the power of the second argument. Special cases:

o If the second argument is positive or negative zero, then the result is 1.0. o If the second argument is 1.0, then the result is the same as the first argument. o If the second argument is NaN, then the result is NaN. o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

**Parameters:**

*base* - the base

*exp* - the exponent

**Returns:**

the base raised to the power of *exp*.

**6.512.3.19 static double decaf::lang::Math::random ()** [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard. The remainder value is mathematically equal to  $f1 - f2 \times n$ , where *n* is the mathematical integer closest to the exact mathematical value of the quotient  $f1/f2$ , and if two mathematical integers are equally close to  $f1/f2$ , then *n* is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN. o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

**Parameters:**

*f1* - the dividend.

*f2* - the divisor

**Returns:**

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

**Returns:**

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

**6.512.3.20 static long long decaf::lang::Math::round (double *value*) [static]**

Returns the closest long long to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 2501)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN\_VALUE** (p. 2433), the result is equal to the value of **Long::MIN\_VALUE** (p. 2433). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX\_VALUE** (p. 2433), the result is equal to the value of **Long::MAX\_VALUE** (p. 2433).

**Parameters:**

*value* - the value to round

**Returns:**

the value of the argument rounded to the nearest integral value.

**6.512.3.21 static int decaf::lang::Math::round (float *value*) [static]**

Returns the closest int to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)**Math.floor** (p. 2501)( a + 0.5f )

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN\_VALUE** (p. 2090), the result is equal to the value of **Integer::MIN\_VALUE** (p. 2090). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX\_VALUE** (p. 2090), the result is equal to the value of **Integer::MAX\_VALUE** (p. 2090).

**Parameters:**

*value* - the value to round

**Returns:**

the value of the argument rounded to the nearest integral value.

**6.512.3.22 static double decaf::lang::Math::signum (double *value*) [static]**

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

**Parameters:**

*value* - the floating-point value whose signum is to be returned

**Returns:**

the signum function of the argument

**6.512.3.23 static float decaf::lang::Math::signum (float *value*) [static]**

Returns Euler's number e raised to the power of a double value. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

**Parameters:**

*value* - the exponent to raise e to

**Returns:**

the value  $e^x$ , where e is the base of the natural logarithms. Returns  $e^x - 1$ . Note that for values of x near 0, the exact sum of  $\expm1(x) + 1$  is much closer to the true result of  $e^x$  than  $\exp(x)$ . Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the value to raise  $e^x - 1$

**Returns:**

the value  $e^x - 1$ . Returns  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

**Parameters:**

*x* - an argument

*y* - another argument

**Returns:**

the  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0 if the argument is greater than zero, -1.0 if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

**Parameters:**

*value* - the floating-point value whose signum is to be returned

**Returns:**

the signum function of the argument

**6.512.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]**

Returns the arc cosine of an angle, in the range of 0.0 through pi. Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

**Parameters:**

*value* - the value to return the arc cosine of.

**Returns:**

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the value to return the arc cosine of.

**Returns:**

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the value to return the arc cosine of.

**Returns:**

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3\*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3\*pi/4.

**Parameters:**

*y* - the ordinate coordinate  
*x* - the abscissa coordinate

**Returns:**

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, cbrt(-x) == -cbrt(x); that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the double to compute the cube root of

**Returns:**

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

**Parameters:**

*value* - an value in radians

**Returns:**

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of  $x$  is defined to be  $(e^x + e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

**Parameters:**

*value* - the number whose hyperbolic cosine is to be found

**Returns:**

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the number whose sin is to be found

**Returns:**

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of  $x$  is defined to be  $(e^x - e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the number whose hyperbolic sin is to be found

**Returns:**

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters:**

*value* - the number whose tangent is to be found

**Returns:**

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of  $x$  is defined to be  $(e^x - e^{-x})/(e^x + e^{-x})$ , in other words,  $\sinh(x)/\cosh(x)$ . Note that the absolute value of the exact tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

**Parameters:**

*value* - the number whose hyperbolic tangent is to be found

**Returns:**

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

**Parameters:**

*value* - the value to find the square root of  
*the* square root of the argument.

**6.512.3.25** `static double decaf::lang::Math::toDegrees (double angrad)` [inline, static]

Returns the measure in degrees of the supplied radian angle.

**Parameters:**

*angrad* - an angle in radians

**Returns:**

the degree measure of the angle.

**6.512.3.26** `static double decaf::lang::Math::toRadians (double angdeg)` [inline, static]

Returns the measure in radians of the supplied degree angle.

**Parameters:**

*angdeg* - an angle in degrees

**Returns:**

the radian measure of the angle.

## 6.512.4 Field Documentation

**6.512.4.1** `const double decaf::lang::Math::E` [static]

**6.512.4.2** `const double decaf::lang::Math::PI` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

## 6.513 activemq::util::MemoryUsage Class Reference

#include <src/main/activemq/util/MemoryUsage.h> Inheritance diagram for activemq::util::MemoryUsage:

### Public Member Functions

- **MemoryUsage** ()  
*Default Constructor.*
- **MemoryUsage** (unsigned long long limit)  
*Creates an instance of an **Usage** (p. 3964) monitor with a set limit.*
- virtual ~**MemoryUsage** ()
- virtual void **waitForSpace** ()  
*Waits forever for more space to be returned to this **Usage** (p. 3964) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)  
*Waits for more space to be returned to this **Usage** (p. 3964) Manager, times out when the given time span in milliseconds elapses.*
- virtual void **enqueueUsage** (unsigned long long value)  
*Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void **increaseUsage** (unsigned long long value)  
*Increases the usage by the value amount.*
- virtual void **decreaseUsage** (unsigned long long value)  
*Decreases the usage by the value amount.*
- virtual bool **isFull** () const  
*Returns true if this **Usage** (p. 3964) instance is full, i.e.*
- unsigned long long **getUsage** () const  
*Gets the current usage amount.*
- void **setUsage** (unsigned long long usage)  
*Sets the current usage amount.*
- unsigned long long **getLimit** () const  
*Gets the current limit amount.*
- void **setLimit** (unsigned long long limit)  
*Sets the current limit amount.*



## 6.513.1 Constructor & Destructor Documentation

### 6.513.1.1 activemq::util::MemoryUsage::MemoryUsage ()

Default Constructor.

### 6.513.1.2 activemq::util::MemoryUsage::MemoryUsage (unsigned long long *limit*)

Creates an instance of an **Usage** (p. 3964) monitor with a set limit.

#### Parameters:

*limit* - amount of memory this manager allows.

### 6.513.1.3 virtual activemq::util::MemoryUsage::~~MemoryUsage () [virtual]

## 6.513.2 Member Function Documentation

### 6.513.2.1 virtual void activemq::util::MemoryUsage::decreaseUsage (unsigned long long *value*) [virtual]

Decreases the usage by the value amount.

#### Parameters:

*value* Amount of space to return to the pool

Implements **activemq::util::Usage** (p. 3964).

### 6.513.2.2 virtual void activemq::util::MemoryUsage::enqueueUsage (unsigned long long *value*) [inline, virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

#### Parameters:

*value* Amount of usage in bytes to add.

Implements **activemq::util::Usage** (p. 3964).

### 6.513.2.3 unsigned long long activemq::util::MemoryUsage::getLimit () const [inline]

Gets the current limit amount.

#### Returns:

the amount that can be used before full.

**6.513.2.4**    `unsigned long long activemq::util::MemoryUsage::getUsage () const`  
                  `[inline]`

Gets the current usage amount.

**Returns:**

the amount of bytes currently used.

**6.513.2.5**    `virtual void activemq::util::MemoryUsage::increaseUsage (unsigned long`  
                  `long value) [virtual]`

Increases the usage by the value amount.

**Parameters:**

*value* Amount of usage to add.

Implements `activemq::util::Usage` (p. 3965).

**6.513.2.6**    `virtual bool activemq::util::MemoryUsage::isFull () const [virtual]`

Returns true if this `Usage` (p. 3964) instance is full, i.e. `Usage` (p. 3964)  $\geq 100\%$

Implements `activemq::util::Usage` (p. 3965).

**6.513.2.7**    `void activemq::util::MemoryUsage::setLimit (unsigned long long limit)`  
                  `[inline]`

Sets the current limit amount.

**Parameters:**

*limit* - The amount that can be used before full.

**6.513.2.8**    `void activemq::util::MemoryUsage::setUsage (unsigned long long usage)`  
                  `[inline]`

Sets the current usage amount.

**Parameters:**

*usage* - The amount to tag as used.

**6.513.2.9**    `virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int`  
                  `timeout) [virtual]`

Waits for more space to be returned to this `Usage` (p. 3964) Manager, times out when the given time span in milliseconds elapses.

**Parameters:**

*timeout* The time to wait for more space.

Implements `activemq::util::Usage` (p. 3965).

**6.513.2.10 virtual void activemq::util::MemoryUsage::waitForSpace () [virtual]**

Waits forever for more space to be returned to this **Usage** (p. 3964) Manager.

Implements **activemq::util::Usage** (p. 3965).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

## 6.514 activemq::commands::Message Class Reference

#include <src/main/activemq/commands/Message.h> Inheritance diagram for activemq::commands::Message:

### Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Message \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*
- virtual bool **isMarshalAware** () const  
*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*
- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)  
*Sets the Acknowledgment Handler that this **Message** (p. 2516) will use when the Acknowledge method is called.*
- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const  
*Gets the Acknowledgment Handler that this **Message** (p. 2516) will use when the Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** \*connection)

*Sets the `ActiveMQConnection` instance that this `Command` (p. 1194) was created from when the session create methods are called to create a `Message` (p. 2516).*

- **core::ActiveMQConnection \* getConnection () const**  
*Gets the `ActiveMQConnection` instance that this `Command` (p. 1194) was created from when the session create methods are called to create a `Message` (p. 2516).*
- virtual unsigned int **getSize () const**  
*Returns the Size of this message in Bytes.*
- virtual bool **isExpired () const**  
*Returns if this message has expired, meaning that its Expiration time has elapsed.*
- virtual void **onSend ()**  
*Allows derived `Message` (p. 2516) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap & getMessageProperties ()**  
*Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.*
- const **util::PrimitiveMap & getMessageProperties () const**
- bool **isReadOnlyProperties () const**  
*Returns if the `Message` (p. 2516) Properties Are Read Only.*
- void **setReadOnlyProperties (bool value)**  
*Set the Read Only State of the `Message` (p. 2516) Properties.*
- bool **isReadOnlyBody () const**  
*Returns if the `Message` (p. 2516) Body is Read Only.*
- void **setReadOnlyBody (bool value)**  
*Set the Read Only State of the `Message` (p. 2516) Content.*
- virtual const **Pointer< ProducerId > & getProducerId () const**
- virtual **Pointer< ProducerId > & getProducerId ()**
- virtual void **setProducerId (const Pointer< ProducerId > &producerId)**
- virtual const **Pointer< ActiveMQDestination > & getDestination () const**
- virtual **Pointer< ActiveMQDestination > & getDestination ()**
- virtual void **setDestination (const Pointer< ActiveMQDestination > &destination)**
- virtual const **Pointer< TransactionId > & getTransactionId () const**
- virtual **Pointer< TransactionId > & getTransactionId ()**
- virtual void **setTransactionId (const Pointer< TransactionId > &transactionId)**
- virtual const **Pointer< ActiveMQDestination > & getOriginalDestination () const**
- virtual **Pointer< ActiveMQDestination > & getOriginalDestination ()**
- virtual void **setOriginalDestination (const Pointer< ActiveMQDestination > &originalDestination)**
- virtual const **Pointer< MessageId > & getMessageId () const**
- virtual **Pointer< MessageId > & getMessageId ()**
- virtual void **setMessageId (const Pointer< MessageId > &messageId)**
- virtual const **Pointer< TransactionId > & getOriginalTransactionId () const**
- virtual **Pointer< TransactionId > & getOriginalTransactionId ()**

- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &**groupID**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**dataStructure**)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const

- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &userID)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool recievedByDFBridge)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool droppable)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &cluster)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long brokerInTime)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long brokerOutTime)
- virtual bool **isMessage** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID \_MESSAGE** = 0

## Protected Attributes

- **core::ActiveMQConnection** \* connection
- **Pointer**< **ProducerId** > producerId
- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **TransactionId** > transactionId
- **Pointer**< **ActiveMQDestination** > originalDestination
- **Pointer**< **MessageId** > messageId
- **Pointer**< **TransactionId** > originalTransactionId
- std::string **groupID**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer**< **ActiveMQDestination** > replyTo
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer**< **DataStructure** > **dataStructure**
- **Pointer**< **ConsumerId** > **targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**

- long long **arrival**
- std::string **userID**
- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< decaf::lang::Pointer< BrokerId > > **cluster**
- long long **brokerInTime**
- long long **brokerOutTime**

## Static Protected Attributes

- static const unsigned int **DEFAULT\_MESSAGE\_SIZE** = 1024

### 6.514.1 Constructor & Destructor Documentation

**6.514.1.1** `activemq::commands::Message::Message ()`

**6.514.1.2** `virtual activemq::commands::Message::~Message () [virtual]`

### 6.514.2 Member Function Documentation

**6.514.2.1** `virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw ( decaf::io::IOException ) [virtual]`

Called after unmarshaling is started to cleanup the object being unmarshaled.

#### Parameters:

*wireFormat* - the **wireformat** (p. 105) object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p. 831).

**6.514.2.2** `virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw ( decaf::io::IOException ) [virtual]`

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

#### Parameters:

*wireFormat* - the **wireformat** (p. 105) controller

Reimplemented from **activemq::commands::BaseDataStructure** (p. 831).

**6.514.2.3** `virtual Message* activemq::commands::Message::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.



**Returns:**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 205), **activemq::commands::ActiveMQBytesMessage** (p. 236), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 540), and **activemq::commands::ActiveMQTextMessage** (p. 664).

#### 6.514.2.4 virtual void activemq::commands::Message::copyDataStructure (const DataStructure \* *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 205), **activemq::commands::ActiveMQBytesMessage** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 540), and **activemq::commands::ActiveMQTextMessage** (p. 664).

#### 6.514.2.5 virtual bool activemq::commands::Message::equals (const DataStructure \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 206), **activemq::commands::ActiveMQBytesMessage** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 429), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 541), **activemq::commands::ActiveMQTextMessage** (p. 664), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 429), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 429), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 429).

**6.514.2.6** `virtual Pointer<core::ActiveMQAckHandler>  
activemq::commands::Message::getAckHandler () const [inline,  
virtual]`

Gets the Acknowledgment Handler that this **Message** (p. 2516) will use when the Acknowledge method is called.

**Returns:**

handler ActiveMQAckHandler to call or NULL if not set

**6.514.2.7** `virtual long long activemq::commands::Message::getArrival () const  
[virtual]`

**6.514.2.8** `virtual long long activemq::commands::Message::getBrokerInTime ()  
const [virtual]`

**6.514.2.9** `virtual long long activemq::commands::Message::getBrokerOutTime ()  
const [virtual]`

**6.514.2.10** `virtual std::vector< decaf::lang::Pointer<BrokerId> >&  
activemq::commands::Message::getBrokerPath () [virtual]`

**6.514.2.11** `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&  
activemq::commands::Message::getBrokerPath () const [virtual]`

**6.514.2.12** `virtual std::vector< decaf::lang::Pointer<BrokerId> >&  
activemq::commands::Message::getCluster () [virtual]`

**6.514.2.13** `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&  
activemq::commands::Message::getCluster () const [virtual]`

**6.514.2.14** `core::ActiveMQConnection* ac-  
tivemq::commands::Message::getConnection () const  
[inline]`

Gets the ActiveMQConnection instance that this **Command** (p. 1194) was created from when the session create methods are called to create a **Message** (p. 2516).

**Returns:**

the ActiveMQConnection parent for this **Message** (p. 2516) or NULL if not set.

- 6.514.2.15 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent ()` [virtual]
- 6.514.2.16 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const` [virtual]
- 6.514.2.17 `virtual std::string& activemq::commands::Message::getCorrelationId ()` [virtual]
- 6.514.2.18 `virtual const std::string& activemq::commands::Message::getCorrelationId () const` [virtual]
- 6.514.2.19 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ()` [virtual]
- 6.514.2.20 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const` [virtual]
- 6.514.2.21 `virtual unsigned char activemq::commands::Message::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 206), **activemq::commands::ActiveMQBytesMessage** (p. 238), **activemq::commands::ActiveMQMapMessage** (p. 366), **activemq::commands::ActiveMQMessage** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 445), **activemq::commands::ActiveMQStreamMessage** (p. 541), and **activemq::commands::ActiveMQTextMessage** (p. 664).

- 6.514.2.22 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ()` [virtual]
- 6.514.2.23 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const` [virtual]
- 6.514.2.24 `virtual long long activemq::commands::Message::getExpiration () const` [virtual]
- 6.514.2.25 `virtual std::string& activemq::commands::Message::getGroupID ()` [virtual]
- 6.514.2.26 `virtual const std::string& activemq::commands::Message::getGroupID () const` [virtual]
- 6.514.2.27 `virtual int activemq::commands::Message::getGroupSequence () const` [virtual]
- 6.514.2.28 `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties ()` [virtual]
- 6.514.2.29 `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () const` [virtual]
- 6.514.2.30 `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId ()` [virtual]
- 6.514.2.31 `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const` [virtual]
- 6.514.2.32 `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties () const` [inline]
- 6.514.2.33 `util::PrimitiveMap& activemq::commands::Message::getMessageProperties ()` [inline]

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

**Returns:**

a reference to the Primitive Map that holds message properties.

- 6.514.2.34** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ()`  
[virtual]
- 6.514.2.35** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () const`  
[virtual]
- 6.514.2.36** `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ()`  
[virtual]
- 6.514.2.37** `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () const`  
[virtual]
- 6.514.2.38** `virtual unsigned char activemq::commands::Message::getPriority () const` [virtual]
- 6.514.2.39** `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ()`  
[virtual]
- 6.514.2.40** `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId () const`  
[virtual]
- 6.514.2.41** `virtual int activemq::commands::Message::getRedeliveryCounter () const` [virtual]
- 6.514.2.42** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ()` [virtual]
- 6.514.2.43** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () const` [virtual]
- 6.514.2.44** `virtual unsigned int activemq::commands::Message::getSize () const`  
[virtual]

Returns the Size of this message in Bytes.

**Returns:**

number of bytes this message equates to.

Reimplemented in `activemq::commands::ActiveMQTextMessage` (p. 665).

- 6.514.2.45** `virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ()`  
[virtual]
- 6.514.2.46** `virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ()`  
`const` [virtual]
- 6.514.2.47** `virtual long long activemq::commands::Message::getTimestamp ()` `const`  
[virtual]
- 6.514.2.48** `virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId ()`  
[virtual]
- 6.514.2.49** `virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId ()` `const`  
[virtual]
- 6.514.2.50** `virtual std::string& activemq::commands::Message::getType ()`  
[virtual]
- 6.514.2.51** `virtual const std::string& activemq::commands::Message::getType ()` `const` [virtual]
- 6.514.2.52** `virtual std::string& activemq::commands::Message::getUserID ()`  
[virtual]
- 6.514.2.53** `virtual const std::string& activemq::commands::Message::getUserID ()` `const` [virtual]
- 6.514.2.54** `virtual bool activemq::commands::Message::isCompressed ()` `const`  
[virtual]
- 6.514.2.55** `virtual bool activemq::commands::Message::isDroppable ()` `const`  
[virtual]
- 6.514.2.56** `virtual bool activemq::commands::Message::isExpired ()` `const`  
[virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

**Returns:**

true if message is expired.

- 6.514.2.57** `virtual bool activemq::commands::Message::isMarshalAware ()` `const`  
[inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

**Returns:**

boolean indicating desire to be in marshaling stages

Reimplemented from `activemq::commands::BaseDataStructure` (p. 832).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 368).

**6.514.2.58** `virtual bool activemq::commands::Message::isMessage () const`  
[inline, virtual]

**Returns:**

an answer of true to the `isMessage()` (p. 2527) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 761).

**6.514.2.59** `virtual bool activemq::commands::Message::isPersistent () const`  
[virtual]

**6.514.2.60** `bool activemq::commands::Message::isReadOnlyBody () const` [inline]

Returns if the `Message` (p. 2516) Body is Read Only.

**Returns:**

true if `Message` (p. 2516) Content is Read Only.

**6.514.2.61** `bool activemq::commands::Message::isReadOnlyProperties () const`  
[inline]

Returns if the `Message` (p. 2516) Properties Are Read Only.

**Returns:**

true if `Message` (p. 2516) Properties are Read Only.

**6.514.2.62** `virtual bool activemq::commands::Message::isRecievedByDFBridge ()`  
`const` [virtual]

**6.514.2.63** `virtual void activemq::commands::Message::onSend ()` [inline,  
virtual]

Allows derived `Message` (p. 2516) classes to perform tasks before a message is sent.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 238),  
`activemq::commands::ActiveMQMessageTemplate< T >` (p. 436),  
`activemq::commands::ActiveMQStreamMessage` (p. 541),  
`activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 436),  
`activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 436),  
`activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 436),  
`activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 436),  
`activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 436), and  
`activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 436).

**6.514.2.64** `virtual void activemq::commands::Message::setAckHandler (const Pointer< core::ActiveMQAckHandler > & handler) [inline, virtual]`

Sets the Acknowledgment Handler that this **Message** (p. 2516) will use when the Acknowledge method is called.

**Parameters:**

*handler* ActiveMQAckHandler to call

**6.514.2.65** `virtual void activemq::commands::Message::setArrival (long long arrival) [virtual]`

**6.514.2.66** `virtual void activemq::commands::Message::setBrokerInTime (long long brokerInTime) [virtual]`

**6.514.2.67** `virtual void activemq::commands::Message::setBrokerOutTime (long long brokerOutTime) [virtual]`

**6.514.2.68** `virtual void activemq::commands::Message::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath) [virtual]`

**6.514.2.69** `virtual void activemq::commands::Message::setCluster (const std::vector< decaf::lang::Pointer< BrokerId > > & cluster) [virtual]`

**6.514.2.70** `virtual void activemq::commands::Message::setCompressed (bool compressed) [virtual]`

**6.514.2.71** `void activemq::commands::Message::setConnection (core::ActiveMQConnection * connection) [inline]`

Sets the ActiveMQConnection instance that this **Command** (p. 1194) was created from when the session create methods are called to create a **Message** (p. 2516).

**Parameters:**

*handler* ActiveMQConnection parent for this message



- 6.514.2.72 virtual void activemq::commands::Message::setContent (const std::vector< unsigned char > & *content*) [virtual]
- 6.514.2.73 virtual void activemq::commands::Message::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.514.2.74 virtual void activemq::commands::Message::setDataStructure (const Pointer< DataStructure > & *dataStructure*) [virtual]
- 6.514.2.75 virtual void activemq::commands::Message::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.514.2.76 virtual void activemq::commands::Message::setDroppable (bool *droppable*) [virtual]
- 6.514.2.77 virtual void activemq::commands::Message::setExpiration (long long *expiration*) [virtual]
- 6.514.2.78 virtual void activemq::commands::Message::setGroupID (const std::string & *groupID*) [virtual]
- 6.514.2.79 virtual void activemq::commands::Message::setGroupSequence (int *groupSequence*) [virtual]
- 6.514.2.80 virtual void activemq::commands::Message::setMarshaledProperties (const std::vector< unsigned char > & *marshalledProperties*) [virtual]
- 6.514.2.81 virtual void activemq::commands::Message::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.514.2.82 virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]
- 6.514.2.83 virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & *originalTransactionId*) [virtual]
- 6.514.2.84 virtual void activemq::commands::Message::setPersistent (bool *persistent*) [virtual]
- 6.514.2.85 virtual void activemq::commands::Message::setPriority (unsigned char *priority*) [virtual]
- 6.514.2.86 virtual void activemq::commands::Message::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]
- 6.514.2.87 void activemq::commands::Message::setReadOnlyBody (bool *value*) [inline]

Set the Read Only State of the Message (p. 2516) Content.

#### Parameters:

*value* - true if Content should be read only.

**6.514.2.88** `void activemq::commands::Message::setReadOnlyProperties (bool value) [inline]`

Set the Read Only State of the **Message** (p. 2516) Properties.

**Parameters:**

*value* - true if Properties should be read only.

**6.514.2.89** `virtual void activemq::commands::Message::setRecievedByDFBridge (bool recievedByDFBridge) [virtual]`

**6.514.2.90** `virtual void activemq::commands::Message::setRedeliveryCounter (int redeliveryCounter) [virtual]`

**6.514.2.91** `virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & replyTo) [virtual]`

**6.514.2.92** `virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & targetConsumerId) [virtual]`

**6.514.2.93** `virtual void activemq::commands::Message::setTimestamp (long long timestamp) [virtual]`

**6.514.2.94** `virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`

**6.514.2.95** `virtual void activemq::commands::Message::setType (const std::string & type) [virtual]`

**6.514.2.96** `virtual void activemq::commands::Message::setUserID (const std::string & userID) [virtual]`

**6.514.2.97** `virtual std::string activemq::commands::Message::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 208), **activemq::commands::ActiveMQBytesMessage** (p. 244), **activemq::commands::ActiveMQMapMessage** (p. 372), **activemq::commands::ActiveMQMessage** (p. 400), **activemq::commands::ActiveMQObjectMessage** (p. 445), **activemq::commands::ActiveMQStreamMessage** (p. 547), and **activemq::commands::ActiveMQTextMessage** (p. 666).

**6.514.2.98**    `virtual Pointer<Command> activemq::commands::Message::visit  
(activemq::state::CommandVisitor * visitor) throw (  
exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.514.3 Field Documentation

- 6.514.3.1 `long long activemq::commands::Message::arrival` [protected]
- 6.514.3.2 `long long activemq::commands::Message::brokerInTime` [protected]
- 6.514.3.3 `long long activemq::commands::Message::brokerOutTime` [protected]
- 6.514.3.4 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::Message::brokerPath` [protected]
- 6.514.3.5 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::Message::cluster` [protected]
- 6.514.3.6 `bool activemq::commands::Message::compressed` [protected]
- 6.514.3.7 `core::ActiveMQConnection* activemq::commands::Message::connection`  
[protected]
- 6.514.3.8 `std::vector<unsigned char> activemq::commands::Message::content`  
[protected]
- 6.514.3.9 `std::string activemq::commands::Message::correlationId` [protected]
- 6.514.3.10 `Pointer<DataStructure> activemq::commands::Message::dataStructure`  
[protected]
- 6.514.3.11 `const unsigned int activemq::commands::Message::DEFAULT_-`  
`MESSAGE_SIZE = 1024` [static, protected]
- 6.514.3.12 `Pointer<ActiveMQDestination> ac-`  
`tivemq::commands::Message::destination` [protected]
- 6.514.3.13 `bool activemq::commands::Message::droppable` [protected]
- 6.514.3.14 `long long activemq::commands::Message::expiration` [protected]
- 6.514.3.15 `std::string activemq::commands::Message::groupID` [protected]
- 6.514.3.16 `int activemq::commands::Message::groupSequence` [protected]
- 6.514.3.17 `const unsigned char activemq::commands::Message::ID_MESSAGE = 0`  
[static]
- 6.514.3.18 `std::vector<unsigned char> ac-`  
`tivemq::commands::Message::marshalledProperties`  
[protected]
- 6.514.3.19 `Pointer<MessageId> activemq::commands::Message::messageId`  
[protected]
- 6.514.3.20 `Pointer<ActiveMQDestination> ac-`  
`tivemq::commands::Message::originalDestination`  
[protected]
- 6.514.3.21 `Pointer<TransactionId> ac-`  
`tivemq::commands::Message::originalTransactionId`  
[protected]
- 6.514.3.22 `bool activemq::commands::Message::persistent` [protected]
- 6.514.3.23 `unsigned char activemq::commands::Message::priority` [protected]

- `src/main/activemq/commands/Message.h`

## 6.515 cms::Message Class Reference

Root of all messages.

#include <src/main/cms/Message.h> Inheritance diagram for cms::Message:

### Public Member Functions

- virtual `~Message ()`
- virtual `Message * clone () const =0`  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void `acknowledge () const =0 throw ( IllegalStateException, CMSEException )`  
*Acknowledges all consumed messages of the session of this consumed message.*
- virtual void `clearBody ()=0 throw ( CMSEException )`  
*Clears out the body of the message.*
- virtual void `clearProperties ()=0 throw ( CMSEException )`  
*Clears out the message body.*
- virtual `std::vector< std::string > getPropertyNames () const =0 throw ( CMSEException )`  
*Retrieves the property names.*
- virtual bool `propertyExists (const std::string &name) const =0 throw ( CMSEException )`  
*Indicates whether or not a given property exists.*
- virtual bool `getBooleanProperty (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )`  
*Gets a boolean property.*
- virtual unsigned char `getByteProperty (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )`  
*Gets a byte property.*
- virtual double `getDoubleProperty (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )`  
*Gets a double property.*
- virtual float `getFloatProperty (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )`  
*Gets a float property.*
- virtual int `getIntProperty (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )`  
*Gets a int property.*

- virtual long long **getLongProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )  
*Gets a string property.*
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a boolean property.*
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a byte property.*
- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a double property.*
- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a float property.*
- virtual void **setIntProperty** (const std::string &name, int value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a int property.*
- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a long property.*
- virtual void **setShortProperty** (const std::string &name, short value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a string property.*
- virtual std::string **getCMSCorrelationID** () const =0 throw ( CMSEException )  
*Gets the correlation ID for the message.*
- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw ( CMSEException )  
*Sets the correlation ID for the message.*

- virtual int **getCMSDeliveryMode** () const =0 throw ( CMSEException )  
*Gets the **DeliveryMode** (p. 1721) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw ( CMSEException )  
*Sets the **DeliveryMode** (p. 1721) for this message.*
- virtual const **Destination** \* **getCMSDestination** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1723) object for this message.*
- virtual void **setCMSDestination** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Sets the **Destination** (p. 1723) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw ( CMSEException )  
*Gets the message's expiration value.*
- virtual void **setCMSExpiration** (long long expireTime)=0 throw ( CMSEException )  
*Sets the message's expiration value.*
- virtual std::string **getCMSMessageID** () const =0 throw ( CMSEException )  
*The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.*
- virtual void **setCMSMessageID** (const std::string &id)=0 throw ( CMSEException )  
*Sets the message ID.*
- virtual int **getCMSPriority** () const =0 throw ( CMSEException )  
*Gets the message priority level.*
- virtual void **setCMSPriority** (int priority)=0 throw ( CMSEException )  
*Sets the Priority Value for this message.*
- virtual bool **getCMSRedelivered** () const =0 throw ( CMSEException )  
*Gets an indication of whether this message is being redelivered.*
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw ( CMSEException )  
*Specifies whether this message is being redelivered.*
- virtual const **cms::Destination** \* **getCMSReplyTo** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1723) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** \*destination)=0 throw ( CMSEException )  
*Sets the **Destination** (p. 1723) object to which a reply to this message should be sent.*
- virtual long long **getCMSTimestamp** () const =0 throw ( CMSEException )  
*Gets the message timestamp.*
- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw ( CMSEException )



*Sets the message timestamp.*

- virtual std::string **getCMSType** () const =0 throw ( CMSEException )

*Gets the message type identifier supplied by the client when the message was sent.*

- virtual void **setCMSType** (const std::string &type)=0 throw ( CMSEException )

*Sets the message type.*

### 6.515.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

#### Message (p. 2534) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2534) Interface definition.

- Stream - A **StreamMessage** (p. 3652) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 1056) type the values written to a **StreamMessage** (p. 3652) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2534) Body.
- Map - A **MapMessage** (p. 2472) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 2472) makes no guarantee on the order of the elements within the **Message** (p. 2534) body.
- Text - A **TextMessage** (p. 3763) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 1056) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

#### Message (p. 2534) Properties

**Message** (p. 2534) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1160). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

|-----

When a **Message** (p. 2534) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSEException** (p. 1160) being thrown.

See also:

JMS API

Since:

1.0

## 6.515.2 Constructor & Destructor Documentation

**6.515.2.1** `virtual cms::Message::~Message () [inline, virtual]`

## 6.515.3 Member Function Documentation

**6.515.3.1** `virtual void cms::Message::acknowledge () const throw (   
 IllegalStateException, CMSEException ) [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message. All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

*IllegalStateException* (p. 1997) - if this method is called on a closed session.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 428), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 428).

**6.515.3.2** `virtual void cms::Message::clearBody () throw ( CMSEException ) [pure virtual]`

Clears out the body of the message. This does not clear the headers or properties.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 236), **activemq::commands::ActiveMQMapMessage** (p. 363), **activemq::commands::ActiveMQStreamMessage** (p. 540), **activemq::commands::ActiveMQTextMessage** (p. 663), **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 428), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>** (p. 428), **activemq::commands::ActiveMQMessageTemplate<cms::Message>** (p. 428), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>** (p. 428), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>** (p. 428), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>** (p. 428).

### 6.515.3.3 virtual void cms::Message::clearProperties () throw ( CMSEException ) [pure virtual]

Clears out the message body. Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 429), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>** (p. 429), **activemq::commands::ActiveMQMessageTemplate<cms::Message>** (p. 429), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>** (p. 429), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>** (p. 429), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>** (p. 429).

### 6.515.3.4 virtual Message\* cms::Message::clone () const [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

**Returns:**

new copy of this message

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 205), **activemq::commands::ActiveMQBytesMessage** (p. 236), **activemq::commands::ActiveMQMapMessage** (p. 363), **activemq::commands::ActiveMQMessage** (p. 398), **activemq::commands::ActiveMQObjectMessage** (p. 444), **activemq::commands::ActiveMQStreamMessage** (p. 540), **activemq::commands::ActiveMQTextMessage** (p. 663), and **cms::BytesMessage** (p. 1059).

**6.515.3.5** `virtual bool cms::Message::getBooleanProperty (const std::string & name) const throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a boolean property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 429), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 429).

**6.515.3.6** `virtual unsigned char cms::Message::getByteProperty (const std::string & name) const throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a byte property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

### 6.515.3.7 virtual std::string cms::Message::getCMSCorrelationID () const throw ( CMSEException ) [pure virtual]

Gets the correlation ID for the message. This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

#### Returns:

string representation of the correlation Id

#### Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

### 6.515.3.8 virtual int cms::Message::getCMSDeliveryMode () const throw ( CMSEException ) [pure virtual]

Gets the **DeliveryMode** (p. 1721) for this message.

#### Returns:

**DeliveryMode** (p. 1721) enumerated value.

#### Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

### 6.515.3.9 virtual const Destination\* cms::Message::getCMSDestination () const throw ( CMSEException ) [pure virtual]

Gets the **Destination** (p. 1723) object for this message. The **CMSDestination** header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its **CMSDestination** value must be equivalent to the value assigned when it was sent.

**Returns:**

**Destination** (p. 1723) object

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 431), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 431).

#### 6.515.3.10 `virtual long long cms::Message::getCMSExpiration () const throw (CMSEException) [pure virtual]`

Gets the message's expiration value. When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

**Returns:**

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 431), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 431).

#### 6.515.3.11 `virtual std::string cms::Message::getCMSMessageID () const throw (CMSEException) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider. When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **MessageProducer.setDisableMessageID** (p. 2730) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

#### Returns:

provider-assigned message id

#### Exceptions:

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 431), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>** (p. 431), **activemq::commands::ActiveMQMessageTemplate<cms::Message>** (p. 431), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>** (p. 431), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>** (p. 431), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>** (p. 431).

#### 6.515.3.12 virtual int cms::Message::getCMSPriority () const throw (CMSException) [pure virtual]

Gets the message priority level. The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

#### Returns:

priority value

#### Exceptions:

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 432), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>** (p. 432), **activemq::commands::ActiveMQMessageTemplate<cms::Message>** (p. 432), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>** (p. 432), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>** (p. 432), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>** (p. 432).

### 6.515.3.13 `virtual bool cms::Message::getCMSRedelivered () const throw ( CMSEException ) [pure virtual]`

Gets an indication of whether this message is being redelivered. If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

#### Returns:

true if this message is being redelivered

#### Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 432), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 432).

### 6.515.3.14 `virtual const cms::Destination* cms::Message::getCMSReplyTo () const throw ( CMSEException ) [pure virtual]`

Gets the **Destination** (p. 1723) object to which a reply to this message should be sent.

#### Returns:

**Destination** (p. 1723) to which to send a response to this message

#### Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 432), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 432).

### 6.515.3.15 `virtual long long cms::Message::getCMSTimestamp () const throw ( CMSEException ) [pure virtual]`

Gets the message timestamp. The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.



Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the `MessageProducer.setDisableMessageTimestamp` method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

**Returns:**

the message timestamp

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 432), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 432), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 432), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 432), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 432), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 432).

**6.515.3.16** `virtual std::string cms::Message::getCMSType () const throw ( CMSException ) [pure virtual]`

Gets the message type identifier supplied by the client when the message was sent.

**Returns:**

the message type

**See also:**

`setCMSType` (p. 2554)

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 433), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 433).

**6.515.3.17** `virtual double cms::Message::getDoubleProperty (const std::string & name) const throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a double property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 433), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 433).

**6.515.3.18** `virtual float cms::Message::getFloatProperty (const std::string & name)  
const throw ( MessageFormatException, CMSEException ) [pure  
virtual]`

Gets a float property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 433), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 433).

**6.515.3.19** `virtual int cms::Message::getIntProperty (const std::string & name)  
const throw ( MessageFormatException, CMSEException ) [pure  
virtual]`

Gets a int property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 434), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 434).

**6.515.3.20** `virtual long long cms::Message::getLongProperty (const std::string & name) const throw ( MessageFormatException, CMSEException )` [pure virtual]

Gets a long property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 434), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 434), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 434).

**6.515.3.21** `virtual std::vector<std::string> cms::Message::getPropertyNames () const throw ( CMSEException )` [pure virtual]

Retrieves the property names.

**Returns:**

The complete set of property names currently in this message.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 435), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 435).

**6.515.3.22** `virtual short cms::Message::getShortProperty (const std::string & name) const throw ( MessageFormatException, CMSEException )` [pure virtual]

Gets a short property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 435), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 435).

**6.515.3.23** `virtual std::string cms::Message::getStringProperty (const std::string & name) const throw ( MessageFormatException, CMSEException )` [pure virtual]

Gets a string property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) if the property does not exist.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 435), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 435), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 435).

**6.515.3.24** `virtual bool cms::Message::propertyExists (const std::string & name) const throw ( CMSException )` [pure virtual]

Indicates whether or not a given property exists.

**Parameters:**

*name* The name of the property to look up.

**Returns:**

True if the property exists in this message.

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 436), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 436), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 436), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 436), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 436), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 436).

**6.515.3.25** `virtual void cms::Message::setBooleanProperty (const std::string & name, bool value) throw ( MessageNotWriteableException, CMSException )` [pure virtual]

Sets a boolean property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 436), `activemq::commands::ActiveMQMessageTemplate<`

`cms::MapMessage` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::Message` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::StreamMessage` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::TextMessage` > (p. 436), and `activemq::commands::ActiveMQMessageTemplate<`  
`cms::ObjectMessage` > (p. 436).

**6.515.3.26** `virtual void cms::Message::setByteProperty (const std::string & name, unsigned char value) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a byte property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<`  
`cms::BytesMessage` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::MapMessage` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::Message` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::StreamMessage` > (p. 436), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::TextMessage` > (p. 436), and `activemq::commands::ActiveMQMessageTemplate<`  
`cms::ObjectMessage` > (p. 436).

**6.515.3.27** `virtual void cms::Message::setCMSCorrelationID (const std::string & correlationId) throw ( CMSException ) [pure virtual]`

Sets the correlation ID for the message. A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

**Parameters:**

*correlationId* The message ID of a message being referred to.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 437), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 437).

### 6.515.3.28 virtual void cms::Message::setCMSDeliveryMode (int *mode*) throw ( CMSEException ) [pure virtual]

Sets the **DeliveryMode** (p. 1721) for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*mode* **DeliveryMode** (p. 1721) enumerated value.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 437), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 437).

### 6.515.3.29 virtual void cms::Message::setCMSDestination (const Destination \* *destination*) throw ( CMSEException ) [pure virtual]

Sets the **Destination** (p. 1723) object for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*destination* **Destination** (p. 1723) Object

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 437), `activemq::commands::ActiveMQMessageTemplate<`

`cms::MapMessage` > (p. 437), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::Message` > (p. 437), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::StreamMessage` > (p. 437), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::TextMessage` > (p. 437), and `activemq::commands::ActiveMQMessageTemplate<`  
`cms::ObjectMessage` > (p. 437).

**6.515.3.30** `virtual void cms::Message::setCMSExpiration (long long expireTime)  
throw ( CMSException ) [pure virtual]`

Sets the message's expiration value. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*expireTime* the message's expiration time

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<`  
`cms::BytesMessage` > (p. 438), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::MapMessage` > (p. 438), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::Message` > (p. 438), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::StreamMessage` > (p. 438), `activemq::commands::ActiveMQMessageTemplate<`  
`cms::TextMessage` > (p. 438), and `activemq::commands::ActiveMQMessageTemplate<`  
`cms::ObjectMessage` > (p. 438).

**6.515.3.31** `virtual void cms::Message::setCMSMessageID (const std::string & id)  
throw ( CMSException ) [pure virtual]`

Sets the message ID. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*id* the ID of the message

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

**6.515.3.32** `virtual void cms::Message::setCMSPriority (int priority) throw (   
CMSException ) [pure virtual]`

Sets the Priority Value for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*priority* priority value for this message



**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 438), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 438).

### 6.515.3.33 virtual void cms::Message::setCMSRedelivered (bool *redelivered*) throw ( CMSException ) [pure virtual]

Specifies whether this message is being redelivered. This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

**Parameters:**

*redelivered* boolean redelivered value

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 438), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 438), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 438).

### 6.515.3.34 virtual void cms::Message::setCMSReplyTo (const cms::Destination \* *destination*) throw ( CMSException ) [pure virtual]

Sets the **Destination** (p.1723) object to which a reply to this message should be sent. The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 3148) object or a **Topic** (p. 3817) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

**Parameters:**

*destination* **Destination** (p.1723) to which to send a response to this message

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 439), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 439).

**6.515.3.35** `virtual void cms::Message::setCMSTimestamp (long long timeStamp) throw ( CMSEException )` [pure virtual]

Sets the message timestamp. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters:**

*timeStamp* integer time stamp value

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 439), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 439).

**6.515.3.36** `virtual void cms::Message::setCMSType (const std::string & type) throw ( CMSEException )` [pure virtual]

Sets the message type. Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

**Parameters:**

*type* the message type

See also:

`getCMSType` (p. 2545)

Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 439), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 439), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 439).

**6.515.3.37** `virtual void cms::Message::setDoubleProperty (const std::string & name, double value) throw ( MessageNotWriteableException, CMSEException ) [pure virtual]`

Sets a double property.

Parameters:

*name* The name of the property to retrieve.

*value* The value for the named property.

Exceptions:

*CMSEException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 440), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 440).

**6.515.3.38** `virtual void cms::Message::setFloatProperty (const std::string & name, float value) throw ( MessageNotWriteableException, CMSEException ) [pure virtual]`

Sets a float property.

Parameters:

*name* The name of the property to retrieve.

*value* The value for the named property.

Exceptions:

*CMSEException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 440), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 440).

**6.515.3.39** `virtual void cms::Message::setIntProperty (const std::string & name, int value) throw ( MessageNotWriteableException, CMSEException )` [pure virtual]

Sets a int property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 440), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 440), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 440).

**6.515.3.40** `virtual void cms::Message::setLongProperty (const std::string & name, long long value) throw ( MessageNotWriteableException, CMSEException )` [pure virtual]

Sets a long property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**Exceptions:**

*CMSEException* (p. 1160) - if the name is an empty string.

*MessageNotWriteableException* (p. 2724) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 441), `activemq::commands::ActiveMQMessageTemplate<`

**cms::MapMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::Message** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::StreamMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::TextMessage** > (p. 441), and **activemq::commands::ActiveMQMessageTemplate**<  
**cms::ObjectMessage** > (p. 441).

**6.515.3.41** `virtual void cms::Message::setShortProperty (const std::string & name, short value) throw ( MessageNotWriteableException, CMSException )` [pure virtual]

Sets a short property.

#### Parameters:

**name** The name of the property to retrieve.

**value** The value for the named property.

#### Exceptions:

**CMSException** (p. 1160) - if the name is an empty string.

**MessageNotWriteableException** (p. 2724) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**<  
**cms::BytesMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::MapMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::Message** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::StreamMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::TextMessage** > (p. 441), and **activemq::commands::ActiveMQMessageTemplate**<  
**cms::ObjectMessage** > (p. 441).

**6.515.3.42** `virtual void cms::Message::setStringProperty (const std::string & name, const std::string & value) throw ( MessageNotWriteableException, CMSException )` [pure virtual]

Sets a string property.

#### Parameters:

**name** The name of the property to retrieve.

**value** The value for the named property.

#### Exceptions:

**CMSException** (p. 1160) - if the name is an empty string.

**MessageNotWriteableException** (p. 2724) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**<  
**cms::BytesMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::MapMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::Message** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::StreamMessage** > (p. 441), **activemq::commands::ActiveMQMessageTemplate**<  
**cms::TextMessage** > (p. 441), and **activemq::commands::ActiveMQMessageTemplate**<  
**cms::ObjectMessage** > (p. 441).

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

## 6.516 activemq::commands::MessageAck Class Reference

#include <src/main/activemq/commands/MessageAck.h> Inheritance diagram for activemq::commands::MessageAck:

### Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_MESSAGEACK** = 22

## Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ConsumerId** > **consumerId**
- unsigned char **ackType**
- **Pointer**< **MessageId** > **firstMessageId**
- **Pointer**< **MessageId** > **lastMessageId**
- int **messageCount**

## 6.516.1 Constructor & Destructor Documentation

**6.516.1.1** **activemq::commands::MessageAck::MessageAck ()**

**6.516.1.2** **virtual activemq::commands::MessageAck::~~MessageAck ()** [virtual]

## 6.516.2 Member Function Documentation

**6.516.2.1** **virtual MessageAck\* activemq::commands::MessageAck::cloneDataStructure ()**  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.516.2.2** **virtual void activemq::commands::MessageAck::copyDataStructure (const DataStructure \* *src*)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.516.2.3** **virtual bool activemq::commands::MessageAck::equals (const DataStructure \* *value*)** const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.



**Returns:**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.516.2.4** virtual unsigned char activemq::commands::MessageAck::getAckType ()  
const [virtual]

**6.516.2.5** virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ()  
[virtual]

**6.516.2.6** virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () const  
[virtual]

**6.516.2.7** virtual unsigned char activemq::commands::MessageAck::getDataStructureType ()  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.516.2.8** `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ()`  
[virtual]
- 6.516.2.9** `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () const` [virtual]
- 6.516.2.10** `virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ()`  
[virtual]
- 6.516.2.11** `virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ()`  
`const` [virtual]
- 6.516.2.12** `virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ()`  
[virtual]
- 6.516.2.13** `virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ()`  
`const` [virtual]
- 6.516.2.14** `virtual int activemq::commands::MessageAck::getMessageCount () const`  
[virtual]
- 6.516.2.15** `virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ()`  
[virtual]
- 6.516.2.16** `virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () const`  
[virtual]
- 6.516.2.17** `virtual bool activemq::commands::MessageAck::isMessageAck () const`  
[inline, virtual]

**Returns:**

an answer of true to the `isMessageAck()` (p. 2562) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 761).

- 6.516.2.18 `virtual void activemq::commands::MessageAck::setAckType (unsigned char ackType) [virtual]`
- 6.516.2.19 `virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.516.2.20 `virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.516.2.21 `virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & firstMessageId) [virtual]`
- 6.516.2.22 `virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & lastMessageId) [virtual]`
- 6.516.2.23 `virtual void activemq::commands::MessageAck::setMessageCount (int messageCount) [virtual]`
- 6.516.2.24 `virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.516.2.25 `virtual std::string activemq::commands::MessageAck::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.516.2.26 `virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

#### Returns:

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

### 6.516.3 Field Documentation

- 6.516.3.1 `unsigned char activemq::commands::MessageAck::ackType` [protected]
- 6.516.3.2 `Pointer<ConsumerId> activemq::commands::MessageAck::consumerId` [protected]
- 6.516.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination` [protected]
- 6.516.3.4 `Pointer<MessageId> activemq::commands::MessageAck::firstMessageId` [protected]
- 6.516.3.5 `const unsigned char activemq::commands::MessageAck::ID_ - MESSAGEACK = 22` [static]
- 6.516.3.6 `Pointer<MessageId> activemq::commands::MessageAck::lastMessageId` [protected]
- 6.516.3.7 `int activemq::commands::MessageAck::messageCount` [protected]
- 6.516.3.8 `Pointer<TransactionId> activemq::commands::MessageAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageAck.h`

## 6.517 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2565).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.517.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2565).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.517.2 Constructor & Destructor Documentation

**6.517.2.1** `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.517.2.2** `virtual activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.517.3 Member Function Documentation

**6.517.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.517.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.517.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.517.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.517.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.517.3.6** virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.517.3.7** virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h



## 6.518 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2569).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.518.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2569).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.518.2 Constructor & Destructor Documentation

**6.518.2.1** `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.518.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.518.3 Member Function Documentation

**6.518.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.518.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.518.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.518.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.518.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.518.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.518.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h

## 6.519 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2573).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.519.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2573).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.519.2 Constructor & Destructor Documentation

**6.519.2.1** `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.519.2.2** `virtual activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.519.3 Member Function Documentation

**6.519.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.519.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.519.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.519.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.519.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.519.3.6** virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.519.3.7** virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h



## 6.520 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2577).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.520.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2577).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.520.2 Constructor & Destructor Documentation

**6.520.2.1** `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.520.2.2** `virtual activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.520.3 Member Function Documentation

**6.520.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.520.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.520.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.520.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.520.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.520.3.6** virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.520.3.7** virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h

## 6.521 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2581).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.521.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2581).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.521.2 Constructor & Destructor Documentation

**6.521.2.1** `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.521.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.521.3 Member Function Documentation

**6.521.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.521.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.521.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.521.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.521.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.521.3.6** virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.521.3.7** virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h



## 6.522 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2585).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.522.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessageAckMarshaller** (p.2585).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.522.2 Constructor & Destructor Documentation

**6.522.2.1** `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller()` [inline]

**6.522.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

## 6.522.3 Member Function Documentation

**6.522.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.522.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.522.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.522.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.522.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.522.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.522.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h

## 6.523 cms::MessageConsumer Class Reference

A client uses a `MessageConsumer` (p. 2589) to received messages from a destination.

```
#include <src/main/cms/MessageConsumer.h> Inheritance diagram for cms::MessageConsumer:
```

### Public Member Functions

- virtual `~MessageConsumer ()`
- virtual `Message * receive ()=0 throw ( CMSEException )`  
*Synchronously Receive a **Message** (p. 2534).*
- virtual `Message * receive (int millisecs)=0 throw ( CMSEException )`  
*Synchronously Receive a **Message** (p. 2534), time out after defined interval.*
- virtual `Message * receiveNoWait ()=0 throw ( CMSEException )`  
*Receive a **Message** (p. 2534), does not wait if there isn't a new message to read, returns **NULL** if nothing read.*
- virtual void `setMessageListener (MessageListener *listener)=0 throw ( CMSEException )`  
*Sets the **MessageListener** (p. 2692) that this class will send notifs on.*
- virtual `MessageListener * getMessageListener () const =0 throw ( CMSEException )`  
*Gets the **MessageListener** (p. 2692) that this class will send new **Message** (p. 2534) notification events to.*
- virtual std::string `getMessageSelector () const =0 throw ( cms::CMSEException )`  
*Gets this message consumer's message selector expression.*

### 6.523.1 Detailed Description

A client uses a `MessageConsumer` (p. 2589) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a `MessageListener` (p. 2692) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the `MessageListener` (p. 2692)'s `onMessage` method.

When the `MessageConsumer`'s close method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a Null when the close method is called.

See also:

`MessageListener` (p. 2692)

Since:

1.0

## 6.523.2 Constructor & Destructor Documentation

**6.523.2.1** `virtual cms::MessageConsumer::~~MessageConsumer () [inline, virtual]`

## 6.523.3 Member Function Documentation

**6.523.3.1** `virtual MessageListener* cms::MessageConsumer::getMessageListener () const throw ( CMSEException ) [pure virtual]`

Gets the `MessageListener` (p. 2692) that this class will send new `Message` (p. 2534) notification events to.

**Returns:**

The listener of messages received by this consumer

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1072), and `activemq::core::ActiveMQConsumer` (p. 315).

**6.523.3.2** `virtual std::string cms::MessageConsumer::getMessageSelector () const throw ( cms::CMSEException ) [pure virtual]`

Gets this message consumer's message selector expression.

**Returns:**

This Consumer's selector expression or "".

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1072), and `activemq::core::ActiveMQConsumer` (p. 316).

**6.523.3.3** `virtual Message* cms::MessageConsumer::receive (int millisecs) throw ( CMSEException ) [pure virtual]`

Synchronously Receive a `Message` (p. 2534), time out after defined interval. Returns null if nothing read.

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1073), and `activemq::core::ActiveMQConsumer` (p. 317).

#### 6.523.3.4 virtual Message\* cms::MessageConsumer::receive () throw (CMSEException) [pure virtual]

Synchronously Receive a **Message** (p. 2534).

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1073), and `activemq::core::ActiveMQConsumer` (p. 317).

#### 6.523.3.5 virtual Message\* cms::MessageConsumer::receiveNoWait () throw (CMSEException) [pure virtual]

Receive a **Message** (p. 2534), does not wait if there isn't a new message to read, returns NULL if nothing read.

**Returns:**

new message which the caller owns and must delete.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1073), and `activemq::core::ActiveMQConsumer` (p. 317).

#### 6.523.3.6 virtual void cms::MessageConsumer::setMessageListener (MessageListener \* listener) throw (CMSEException) [pure virtual]

Sets the **MessageListener** (p. 2692) that this class will send notifs on.

**Parameters:**

*listener* The listener of messages received by this consumer.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1073), and `activemq::core::ActiveMQConsumer` (p. 318).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageConsumer.h`



## 6.524 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the CmsTemplate (p. 1170).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

### Public Member Functions

- virtual `~MessageCreator()`
- virtual `cms::Message * createMessage (cms::Session *session)=0` throw ( `cms::CMSEException` )

*Creates a message from the given session.*

### 6.524.1 Detailed Description

Creates the user-defined message to be sent by the CmsTemplate (p. 1170).

### 6.524.2 Constructor & Destructor Documentation

**6.524.2.1** virtual `activemq::cmsutil::MessageCreator::~MessageCreator()`  
[inline, virtual]

### 6.524.3 Member Function Documentation

**6.524.3.1** virtual `cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * session)` throw ( `cms::CMSEException` ) [pure virtual]

Creates a message from the given session.

#### Parameters:

*session* the CMS Session

#### Exceptions:

*cms::CMSEException* (p. 1160) if thrown by CMS API methods

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/MessageCreator.h`

## 6.525 activemq::commands::MessageDispatch Class Reference

#include <src/main/activemq/commands/MessageDispatch.h> Inheritance diagram for activemq::commands::MessageDispatch:

### Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatch** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE\_DISPATCH** = 21

## Protected Attributes

- `Pointer< ConsumerId > consumerId`
- `Pointer< ActiveMQDestination > destination`
- `Pointer< Message > message`
- `int redeliveryCounter`

## 6.525.1 Constructor & Destructor Documentation

**6.525.1.1** `activemq::commands::MessageDispatch::MessageDispatch ()`

**6.525.1.2** `virtual activemq::commands::MessageDispatch::~~MessageDispatch ()`  
[virtual]

## 6.525.2 Member Function Documentation

**6.525.2.1** `virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.525.2.2** `virtual void activemq::commands::MessageDispatch::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.525.2.3** `virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.525.2.4** `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ()`  
[virtual]
- 6.525.2.5** `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ()`  
const [virtual]
- 6.525.2.6** `virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType ()` const  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSetructure** (p. 1660) type copy.

Implements **activemq::commands::DataSetructure** (p. 1662).

- 6.525.2.7** `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ()`  
[virtual]
- 6.525.2.8** `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ()` const  
[virtual]
- 6.525.2.9** `virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage ()`  
[virtual]
- 6.525.2.10** `virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage ()`  
const [virtual]
- 6.525.2.11** `virtual int activemq::commands::MessageDispatch::getRedeliveryCounter ()` const  
[virtual]
- 6.525.2.12** `virtual bool activemq::commands::MessageDispatch::isMessageDispatch ()` const [inline, virtual]

**Returns:**

an answer of true to the **isMessageDispatch()** (p. 2596) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 761).

- 6.525.2.13** `virtual void activemq::commands::MessageDispatch::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.525.2.14** `virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.525.2.15** `virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & message) [virtual]`
- 6.525.2.16** `virtual void activemq::commands::MessageDispatch::setRedeliveryCounter (int redeliveryCounter) [virtual]`
- 6.525.2.17** `virtual std::string activemq::commands::MessageDispatch::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.525.2.18** `virtual Pointer<Command> activemq::commands::MessageDispatch::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

#### Returns:

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

### 6.525.3 Field Documentation

- 6.525.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId`  
[protected]
- 6.525.3.2 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination`  
[protected]
- 6.525.3.3 `const unsigned char activemq::commands::MessageDispatch::ID_ -  
MESSAGEDISPATCH = 21` [static]
- 6.525.3.4 `Pointer<Message> activemq::commands::MessageDispatch::message`  
[protected]
- 6.525.3.5 `int activemq::commands::MessageDispatch::redeliveryCounter`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatch.h`

## 6.526 activemq::core::MessageDispatchChannel Class Reference

#include <src/main/activemq/core/MessageDispatchChannel.h> Inheritance diagram for activemq::core::MessageDispatchChannel:

### Public Member Functions

- **MessageDispatchChannel** ()
- virtual **~MessageDispatchChannel** ()
- void **enqueue** (const **Pointer**< **MessageDispatch** > &message)  
*Add a Message to the Channel behind all pending message.*
- void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)  
*Add a message to the front of the Channel.*
- bool **isEmpty** () const
- bool **isClosed** () const
- bool **isRunning** () const
- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw ( exceptions::ActiveMQException )  
*Used to get an enqueued message.*
- **Pointer**< **MessageDispatch** > **dequeueNoWait** ()  
*Used to get an enqueued message if there is one queued right now.*
- **Pointer**< **MessageDispatch** > **peek** () const  
*Peek in the Queue and return the first message in the Channel without removing it from the channel.*
- void **start** ()  
*Starts dispatch of messages from the Channel.*
- void **stop** ()  
*Stops dispatch of message from the Channel.*
- void **close** ()  
*Close this channel no messages will be dispatched after this method is called.*
- void **clear** ()  
*Clear the Channel, all pending messages are removed.*
- int **size** () const
- std::vector< **Pointer**< **MessageDispatch** > > **removeAll** ()  
*Remove all messages that are currently in the Channel and return them as a list of Messages.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

## 6.526.1 Constructor & Destructor Documentation

### 6.526.1.1 activemq::core::MessageDispatchChannel::MessageDispatchChannel ()

### 6.526.1.2 virtual activemq::core::MessageDispatchChannel::~~MessageDispatchChannel () [virtual]

## 6.526.2 Member Function Documentation

### 6.526.2.1 void activemq::core::MessageDispatchChannel::clear ()

Clear the Channel, all pending messages are removed.

### 6.526.2.2 void activemq::core::MessageDispatchChannel::close ()

Close this channel no messages will be dispatched after this method is called.



### 6.526.2.3 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long long timeout) throw ( exceptions::ActiveMQException )`

Used to get an enqueued message. The amount of time this method blocks is based on the timeout value. - if timeout==1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

#### Returns:

null if we timeout or if the consumer is closed.

#### Exceptions:

*ActiveMQException*

### 6.526.2.4 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ()`

Used to get an enqueued message if there is one queued right now. If there is no waiting message then this method returns Null.

#### Returns:

a message if there is one in the queue.

### 6.526.2.5 `void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message)`

Add a Message to the Channel behind all pending message.

#### Parameters:

*message* - The message to add to the Channel.

### 6.526.2.6 `void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message)`

Add a message to the front of the Channel.

#### Parameters:

*message* - The Message to add to the front of the Channel.

### 6.526.2.7 `bool activemq::core::MessageDispatchChannel::isClosed () const [inline]`

#### Returns:

has the Queue been closed.

**6.526.2.8** `bool activemq::core::MessageDispatchChannel::isEmpty () const`**Returns:**

true if there are no messages in the Channel.

**6.526.2.9** `bool activemq::core::MessageDispatchChannel::isRunning () const`  
[inline]**Returns:**

true if the Channel currently running and will dequeue message.

**6.526.2.10** `virtual void activemq::core::MessageDispatchChannel::lock () throw (`  
`decaf::lang::exceptions::RuntimeException )` [inline, virtual]

Locks the object.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3701).

**6.526.2.11** `virtual void activemq::core::MessageDispatchChannel::notify`  
`() throw ( decaf::lang::exceptions::RuntimeException,`  
`decaf::lang::exceptions::IllegalMonitorStateException )` [inline,  
virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads (p. 96).

Implements `decaf::util::concurrent::Synchronizable` (p. 3702).

**6.526.2.12** `virtual void activemq::core::MessageDispatchChannel::notifyAll`  
`() throw ( decaf::lang::exceptions::RuntimeException,`  
`decaf::lang::exceptions::IllegalMonitorStateException )` [inline,  
virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying the waiting **threads** (p. 96).

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.526.2.13** `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek () const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

**Returns:**

a message if there is one in the queue.

**6.526.2.14** `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ()`

Remove all messages that are currently in the Channel and return them as a list of Messages.

**Returns:**

a list of Messages that was previously in the Channel.

**6.526.2.15** `int activemq::core::MessageDispatchChannel::size () const`

**Returns:**

the number of Messages currently in the Channel.

**6.526.2.16** `void activemq::core::MessageDispatchChannel::start ()`

Starts dispatch of messages from the Channel.

**6.526.2.17** `void activemq::core::MessageDispatchChannel::stop ()`

Stops dispatch of message from the Channel.

**6.526.2.18** `virtual bool activemq::core::MessageDispatchChannel::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3704).

**6.526.2.19** `virtual void activemq::core::MessageDispatchChannel::unlock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3705).

**6.526.2.20** `virtual void activemq::core::MessageDispatchChannel::wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

**Exceptions:**

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3707).

**6.526.2.21** `virtual void activemq::core::MessageDispatchChannel::wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

```
6.526.2.22 virtual void activemq::core::MessageDispatchChannel::wait  
() throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- src/main/activemq/core/MessageDispatchChannel.h

## 6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2606).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.527.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2606).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.527.2 Constructor & Destructor Documentation

**6.527.2.1** `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatchMarshaller()` `[inline]`

**6.527.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` `[inline, virtual]`

## 6.527.3 Member Function Documentation

**6.527.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.527.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.527.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.527.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.527.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).



**6.527.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.527.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h

## 6.528 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2610).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.528.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2610).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.528.2 Constructor & Destructor Documentation

**6.528.2.1** `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatchMarshaller()` [inline]

**6.528.2.2** `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` [inline, virtual]

## 6.528.3 Member Function Documentation

**6.528.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.528.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.528.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.528.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.528.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.528.3.6** virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.528.3.7** virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h

## 6.529 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2614).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.529.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2614).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.529.2 Constructor & Destructor Documentation

**6.529.2.1** `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatchMarshaller()` [inline]

**6.529.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` [inline, virtual]

## 6.529.3 Member Function Documentation

**6.529.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.529.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.529.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.529.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.529.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).



**6.529.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.529.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h

## 6.530 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2618).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.530.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2618).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.530.2 Constructor & Destructor Documentation

**6.530.2.1** `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatchMarshaller()` `[inline]`

**6.530.2.2** `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` `[inline, virtual]`

## 6.530.3 Member Function Documentation

**6.530.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.530.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.530.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.530.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.530.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.530.3.6** virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.530.3.7** virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h

## 6.531 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for `MessageDispatchMarshaller` (p.2622).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.531.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for `MessageDispatchMarshaller` (p.2622).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.531.2 Constructor & Destructor Documentation

**6.531.2.1** `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatchMarshaller()` `[inline]`

**6.531.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` `[inline, virtual]`

## 6.531.3 Member Function Documentation

**6.531.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.531.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.531.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.531.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.531.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).



**6.531.3.6** virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.531.3.7** virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h

## 6.532 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2626).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`:

### Public Member Functions

- `MessageDispatchMarshaller ()`
- `virtual ~MessageDispatchMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.532.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageDispatchMarshaller` (p. 2626).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.532.2 Constructor & Destructor Documentation

**6.532.2.1** `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::MessageDispatchMarshaller()` [inline]

**6.532.2.2** `virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::~~MessageDispatchMarshaller()` [inline, virtual]

## 6.532.3 Member Function Documentation

**6.532.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.532.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.532.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.532.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.532.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.532.3.6** virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.532.3.7** virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h

## 6.533 activemq::commands::MessageDispatchNotification Class Reference

#include <src/main/activemq/commands/MessageDispatchNotification.h> Inheritance diagram for activemq::commands::MessageDispatchNotification:

### Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatchNotification \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE\_DISPATCH\_NOTIFICATION** = 90

## Protected Attributes

- `Pointer< ConsumerId > consumerId`
- `Pointer< ActiveMQDestination > destination`
- `long long deliverySequenceId`
- `Pointer< MessageId > messageId`

## 6.533.1 Constructor & Destructor Documentation

**6.533.1.1** `activemq::commands::MessageDispatchNotification::MessageDispatchNotification()`

**6.533.1.2** `virtual`  
`activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification()` [virtual]

## 6.533.2 Member Function Documentation

**6.533.2.1** `virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure() const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.533.2.2** `virtual void activemq::commands::MessageDispatchNotification::copyDataStructure(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.533.2.3** `virtual bool activemq::commands::MessageDispatchNotification::equals(const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.533.2.4 `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ()`  
[virtual]
- 6.533.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ()`  
const [virtual]
- 6.533.2.6 `virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType`  
`() const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.533.2.7 `virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId`  
`() const` [virtual]
- 6.533.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ()`  
[virtual]
- 6.533.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ()`  
const [virtual]
- 6.533.2.10 `virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ()`  
[virtual]
- 6.533.2.11 `virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ()` const  
[virtual]
- 6.533.2.12 `virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification`  
`() const` [inline, virtual]

**Returns:**

an answer of true to the `isMessageDispatchNotification()` (p. 2632) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 761).



- 6.533.2.13** virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.533.2.14** virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long *deliverySequenceId*) [virtual]
- 6.533.2.15** virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.533.2.16** virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.533.2.17** virtual std::string activemq::commands::MessageDispatchNotification::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.533.2.18** virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor \* *visitor*) throw ( exceptions::ActiveMQException ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

### 6.533.3 Field Documentation

- 6.533.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId`  
[protected]
- 6.533.3.2 `long long activemq::commands::MessageDispatchNotification::deliverySequenceId`  
[protected]
- 6.533.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination`  
[protected]
- 6.533.3.4 `const unsigned char activemq::commands::MessageDispatchNotification::ID - MESSAGEDISPATCHNOTIFICATION = 90` [static]
- 6.533.3.5 `Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatchNotification.h`

6.534 ac-

tivemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller

Class Reference

6.534 ~~activemq::wireformat::openwire::marshal::v6::MessageDispatchNo~~<sup>2637</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2635).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.534.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2635). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.534.2 Constructor & Destructor Documentation

**6.534.2.1** `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.534.2.2** `virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.534.3 Member Function Documentation

**6.534.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.534.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.534.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

**6.534 ac-**  
**tivemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**  
**Class Reference** **2639**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
 (p. 794).

**6.534.3.4 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseU**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
 (p. 795).

**6.534.3.5 virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightM**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
 (p. 796).

**6.534.3.6** virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.534.3.7** virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h

6.535 ac-

tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller

Class Reference

6.535 — activemq::wireformat::openwire::marshal::v2::MessageDispatchNo<sup>2641</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2639).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.535.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2639). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.535.2 Constructor & Destructor Documentation

**6.535.2.1** `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.535.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.535.3 Member Function Documentation

**6.535.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.535.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.535.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



**6.535** **ac-**  
**tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**  
**Class Reference** **2643**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 801).

**6.535.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseU**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 802).

**6.535.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 803).

**6.535.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.535.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h

6.536 ac-

tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller

Class Reference

6.536 — activemq::wireformat::openwire::marshal::v3::MessageDispatchNo<sup>2645</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2643).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.536.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2643). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.536.2 Constructor & Destructor Documentation

**6.536.2.1** `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.536.2.2** `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.536.3 Member Function Documentation

**6.536.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.536.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.536.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

**6.536** **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**  
**Class Reference** **2647**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
 (p. 766).

**6.536.3.4** **virtual void** **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseUnmarshal**  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
 (p. 767).

**6.536.3.5** **virtual int** **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal**  
 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
 (p. 768).

**6.536.3.6** virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.536.3.7** virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h

6.537 ac-

tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller

Class Reference

6.537 ~~activemq::wireformat::openwire::marshal::v4::MessageDispatchNo~~ 2649

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2647).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.537.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2647). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.537.2 Constructor & Destructor Documentation

**6.537.2.1** `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.537.2.2** `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.537.3 Member Function Documentation

**6.537.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.537.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.537.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



**6.537 ac-**  
**tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**  
**Class Reference** **2651**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
 (p. 773).

**6.537.3.4 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseU**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
 (p. 774).

**6.537.3.5 virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightM**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
 (p. 775).

**6.537.3.6** virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.537.3.7** virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h

6.538 ac-

tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller

Class Reference

6.538 — activemq::wireformat::openwire::marshal::v1::MessageDispatchNo<sup>2653</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2651).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.538.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2651). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.538.2 Constructor & Destructor Documentation

**6.538.2.1** `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.538.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.538.3 Member Function Documentation

**6.538.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.538.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.538.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

**6.538** **ac-**  
**tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**  
**Class Reference** **2655**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 780).

**6.538.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseU**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker

***dataStructure*** - Object to be marshaled

***dataIn*** - BinaryReader that provides that data source

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 781).

**6.538.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker

***dataStructure*** - Object to be marshaled

***bs*** - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

***IOException*** if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 782).

**6.538.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.538.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h`

6.539 ac-

tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

Class Reference

6.539 — activemq::wireformat::openwire::marshal::v5::MessageDispatchNo<sup>2657</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2655).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller  
diagram for activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.539.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2655). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.539.2 Constructor & Destructor Documentation

**6.539.2.1** `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

**6.539.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

## 6.539.3 Member Function Documentation

**6.539.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.539.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.539.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



**6.539 ac-**  
**tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**  
**Class Reference** **2659**  
 Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
 (p. 787).

**6.539.3.4 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseU**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
 (p. 788).

**6.539.3.5 virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightM**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
 (p. 789).

**6.539.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.539.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h

## 6.540 cms::MessageEnumeration Class Reference

Defines an object that enumerates a collection of Messages.

#include <src/main/cms/MessageEnumeration.h> Inheritance diagram for cms::MessageEnumeration:

### Public Member Functions

- virtual `~MessageEnumeration ()`
- virtual bool `hasMoreMessages ()=0`  
*Returns true if there are more **Message** (p. 2534) in the Browser that can be retrieved via the `nextMessage` method.*
- virtual `cms::Message * nextMessage ()=0` throw ( cms::CMSEException )  
*Returns the Next **Message** (p. 2534) in the **Queue** (p. 3148) if one is present, if no more Message's are available then an Exception is thrown.*

### 6.540.1 Detailed Description

Defines an object that enumerates a collection of Messages. The client calls the `hasMoreMessages` method to determine if a **Message** (p. 2534) is available. If a **Message** (p. 2534) is available the client calls the `nextMessage` method to retrieve that **Message** (p. 2534), calling `nextMessage` when a **Message** (p. 2534) is not available results in an exception.

Since:

2.1

### 6.540.2 Constructor & Destructor Documentation

6.540.2.1 `virtual cms::MessageEnumeration::~~MessageEnumeration ()` [inline, virtual]

### 6.540.3 Member Function Documentation

6.540.3.1 `virtual bool cms::MessageEnumeration::hasMoreMessages ()` [pure virtual]

Returns true if there are more **Message** (p. 2534) in the Browser that can be retrieved via the `nextMessage` method. If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

Returns:

true if more Message's are available in the Browser.

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 489).

**6.540.3.2** `virtual cms::Message* cms::MessageEnumeration::nextMessage () throw ( cms::CMSException ) [pure virtual]`

Returns the Next **Message** (p. 2534) in the **Queue** (p. 3148) if one is present, if no more Message's are available then an Exception is thrown. If a **Message** (p. 2534) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

**Returns:**

The next **Message** (p. 2534) in the **Queue** (p. 3148).

**Exceptions:**

*CMSException* (p. 1160) if no more Message's currently in the **Queue** (p. 3148).

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 489).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEnumeration.h`

## 6.541 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3652) or **BytesMessage** (p. 1056) is being read.

#include <src/main/cms/MessageEOFException.h> Inheritance diagram for cms::MessageEOFException:

### Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

#### 6.541.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3652) or **BytesMessage** (p. 1056) is being read.

Since:

1.3

#### 6.541.2 Constructor & Destructor Documentation

- 6.541.2.1** cms::MessageEOFException::MessageEOFException () throw ()
- 6.541.2.2** cms::MessageEOFException::MessageEOFException (const MessageEOFException & *ex*) throw ()
- 6.541.2.3** cms::MessageEOFException::MessageEOFException (const std::string & *message*, const std::exception \* *cause*) throw ()
- 6.541.2.4** cms::MessageEOFException::MessageEOFException (const std::string & *message*, const std::exception \* *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.541.2.5** virtual cms::MessageEOFException::~~MessageEOFException () throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageEOFException.h

## 6.542 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

#include <src/main/cms/MessageFormatException.h> Inheritance diagram for cms::MessageFormatException:

### Public Member Functions

- **MessageFormatException** () throw ()
- **MessageFormatException** (const **MessageFormatException** &ex) throw ()
- **MessageFormatException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageFormatException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageFormatException** () throw ()

### 6.542.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p. 3659) is used to read a boolean value.

Since:

1.3

### 6.542.2 Constructor & Destructor Documentation

- 6.542.2.1** cms::MessageFormatException::MessageFormatException () throw ()
- 6.542.2.2** cms::MessageFormatException::MessageFormatException (const MessageFormatException & ex) throw ()
- 6.542.2.3** cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception \* cause) throw ()
- 6.542.2.4** cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()
- 6.542.2.5** virtual cms::MessageFormatException::~MessageFormatException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageFormatException.h

## 6.543 activemq::commands::MessageId Class Reference

#include <src/main/activemq/commands/MessageId.h> Inheritance diagram for activemq::commands::MessageId:

### Public Types

- typedef **decaf::lang::PointerComparator**< MessageId > **COMPARATOR**

### Public Member Functions

- **MessageId** ()
- **MessageId** (const **MessageId** &other)
- **MessageId** (const std::string &messageKey)
- **MessageId** (const **Pointer**< **ProducerInfo** > &producerInfo, long long **producerSequenceId**)
- **MessageId** (const **Pointer**< **ProducerId** > &producerId, long long **producerSequenceId**)
- **MessageId** (const std::string &producerId, long long **producerSequenceId**)
- virtual ~**MessageId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- void **setValue** (const std::string &key)
- void **setTextView** (const std::string &key)
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceId** () const
- virtual void **setProducerSequenceId** (long long **producerSequenceId**)
- virtual long long **getBrokerSequenceId** () const
- virtual void **setBrokerSequenceId** (long long **brokerSequenceId**)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const

- virtual bool **operator**== (const **MessageId** &value) const
- virtual bool **operator**< (const **MessageId** &value) const
- **MessageId** & **operator**= (const **MessageId** &other)

## Static Public Attributes

- static const unsigned char **ID\_MESSAGEID** = 110

## Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

## 6.543.1 Member Typedef Documentation

- 6.543.1.1 **typedef** decaf::lang::PointerComparator<MessageId>  
activemq::commands::MessageId::COMPARATOR

## 6.543.2 Constructor & Destructor Documentation

- 6.543.2.1 **activemq::commands::MessageId::MessageId** ()
- 6.543.2.2 **activemq::commands::MessageId::MessageId** (const **MessageId** & *other*)
- 6.543.2.3 **activemq::commands::MessageId::MessageId** (const std::string & *messageKey*)
- 6.543.2.4 **activemq::commands::MessageId::MessageId** (const **Pointer**< **ProducerInfo** > & *producerInfo*, long long *producerSequenceId*)
- 6.543.2.5 **activemq::commands::MessageId::MessageId** (const **Pointer**< **ProducerId** > & *producerId*, long long *producerSequenceId*)
- 6.543.2.6 **activemq::commands::MessageId::MessageId** (const std::string & *producerId*, long long *producerSequenceId*)
- 6.543.2.7 **virtual** **activemq::commands::MessageId::~MessageId** () [virtual]

## 6.543.3 Member Function Documentation

- 6.543.3.1 **virtual** **MessageId\*** **activemq::commands::MessageId::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).



**6.543.3.2** `virtual int activemq::commands::MessageId::compareTo (const MessageId & value) const` [virtual]

**6.543.3.3** `virtual void activemq::commands::MessageId::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

**6.543.3.4** `virtual bool activemq::commands::MessageId::equals (const MessageId & value) const` [virtual]

**6.543.3.5** `virtual bool activemq::commands::MessageId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

**6.543.3.6** `virtual long long activemq::commands::MessageId::getBrokerSequenceId () const` [virtual]

**6.543.3.7** `virtual unsigned char activemq::commands::MessageId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.543.3.8 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ()`  
[virtual]
- 6.543.3.9 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId () const`  
[virtual]
- 6.543.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId ()`  
`const` [virtual]
- 6.543.3.11 `virtual bool activemq::commands::MessageId::operator< (const`  
`MessageId & value) const` [virtual]
- 6.543.3.12 `MessageId& activemq::commands::MessageId::operator= (const`  
`MessageId & other)`
- 6.543.3.13 `virtual bool activemq::commands::MessageId::operator== (const`  
`MessageId & value) const` [virtual]
- 6.543.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId`  
`(long long brokerSequenceId)` [virtual]
- 6.543.3.15 `virtual void activemq::commands::MessageId::setProducerId (const`  
`Pointer< ProducerId > & producerId)` [virtual]
- 6.543.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId`  
`(long long producerSequenceId)` [virtual]
- 6.543.3.17 `void activemq::commands::MessageId::setTextView (const std::string &`  
`key)`
- 6.543.3.18 `void activemq::commands::MessageId::setValue (const std::string & key)`
- 6.543.3.19 `virtual std::string activemq::commands::MessageId::toString () const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p.833).

### 6.543.4 Field Documentation

- 6.543.4.1 `long long activemq::commands::MessageId::brokerSequenceId`  
[protected]
- 6.543.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID`  
`= 110` [static]
- 6.543.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`  
[protected]
- 6.543.4.4 `long long activemq::commands::MessageId::producerSequenceId`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

## 6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2668).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.544.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2668). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.544.2 Constructor & Destructor Documentation

6.544.2.1 `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.544.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.544.3 Member Function Documentation

6.544.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

6.544.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

6.544.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.544.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.544.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.544.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.544.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h

## 6.545 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2672).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.545.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2672). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.545.2 Constructor & Destructor Documentation

**6.545.2.1** `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller()` [inline]

**6.545.2.2** `virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.545.3 Member Function Documentation

**6.545.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.545.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.545.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.545.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.545.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.545.3.6** virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.545.3.7** virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h

## 6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2676).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.546.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2676). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.546.2 Constructor & Destructor Documentation

**6.546.2.1** `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller()` [inline]

**6.546.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.546.3 Member Function Documentation

**6.546.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.546.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.546.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.546.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.546.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.546.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.546.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h

## 6.547 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2680).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.547.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2680). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.547.2 Constructor & Destructor Documentation

**6.547.2.1** `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller()` [inline]

**6.547.2.2** `virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.547.3 Member Function Documentation

**6.547.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.547.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.547.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.547.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.547.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.547.3.6** virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.547.3.7** virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h

## 6.548 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2684).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.548.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2684). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.548.2 Constructor & Destructor Documentation

**6.548.2.1** `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::MessageIdMarshaller()` [inline]

**6.548.2.2** `virtual activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.548.3 Member Function Documentation

**6.548.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.548.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.548.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.548.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.548.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.548.3.6** virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.548.3.7** virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h

## 6.549 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2688).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h> Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller`:

### Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.549.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `MessageIdMarshaller` (p. 2688). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.549.2 Constructor & Destructor Documentation

**6.549.2.1** `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller()` [inline]

**6.549.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

## 6.549.3 Member Function Documentation

**6.549.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.549.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.549.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.549.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.549.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.549.3.6** virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.549.3.7** virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h

## 6.550 cms::MessageListener Class Reference

A `MessageListener` (p. 2692) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

### Public Member Functions

- virtual `~MessageListener ()`
- virtual void `onMessage (const Message *message)=0`

*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2534) types.*

### 6.550.1 Detailed Description

A `MessageListener` (p. 2692) object is used to receive asynchronously delivered messages.

Since:

1.0

### 6.550.2 Constructor & Destructor Documentation

6.550.2.1 virtual `cms::MessageListener::~~MessageListener ()` [inline, virtual]

### 6.550.3 Member Function Documentation

6.550.3.1 virtual void `cms::MessageListener::onMessage (const Message * message)`  
[pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2534) types. a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the `onMessage` function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 2534).

It is considered a programming error for this method to throw an exception.

Parameters:

*message* **Message** (p. 2534) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- `src/main/cms/MessageListener.h`

## 6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2693).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.551.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2693). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.551.2 Constructor & Destructor Documentation

**6.551.2.1** `activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller()` [inline]

**6.551.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.551.3 Member Function Documentation

**6.551.3.1** `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 262), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 387), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 414), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 565), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 680).

**6.551.3.2** `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 415), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 566), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 681).

```
6.551.3.3 virtual int activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 415), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 566), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 681).

**6.551.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 263), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 415), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 566), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 681).

**6.551.3.5** `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).



Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 264), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 416), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 461), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 682).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h`

## 6.552 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2698).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.552.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2698). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.552.2 Constructor & Destructor Documentation

- 6.552.2.1 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller()` [inline]
- 6.552.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.552.3 Member Function Documentation

- 6.552.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 210), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 250), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 553), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 668).

- 6.552.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 251), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 554), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 669).

```
6.552.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat,  commands::DataStructure *
            dataStructure,  utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

***IOException*** if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 211), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 251), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 554), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 669).

**6.552.3.4** virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 211), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 554), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 669).

**6.552.3.5** virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 252), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 449), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 670).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

## 6.553 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2703).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.553.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2703). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.553.2 Constructor & Destructor Documentation

**6.553.2.1** `activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller()` [inline]

**6.553.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.553.3 Member Function Documentation

**6.553.3.1** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 270), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 391), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 569), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 688).

**6.553.3.2** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source



**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 271), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 419), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 570), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 689).

```
6.553.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1
          (OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 271), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 419), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 570), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 689).

**6.553.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 271), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 419), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 570), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 689).

**6.553.3.5** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 272), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 420), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 465), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 690).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

## 6.554 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2708).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.554.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2708). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.554.2 Constructor & Destructor Documentation

- 6.554.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller()` [inline]
- 6.554.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.554.3 Member Function Documentation

- 6.554.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 218), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 258), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 383), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 410), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 561), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 672).

- 6.554.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 259), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 384), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 411), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 562), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 673).

```
6.554.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat,  commands::DataStructure *
            dataStructure,  utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

***wireFormat*** - describes the wire format of the broker

***dataStructure*** - Object to be marshaled

***bs*** - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

***IOException*** if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 259), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 384), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 411), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 562), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 673).

**6.554.3.4** virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 219), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 259), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 384), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 411), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 456), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 562), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 673).

**6.554.3.5** virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 220), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 260), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 412), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 457), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 674).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h`



## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2713).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.555.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2713). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.555.2 Constructor & Destructor Documentation

**6.555.2.1** `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller()` [inline]

**6.555.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.555.3 Member Function Documentation

**6.555.3.1** `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 254), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 379), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 406), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 557), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 676).

**6.555.3.2** `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 380), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 407), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 558), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 677).

```
6.555.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
          (OpenWireFormat * wireFormat, commands::DataStructure *
           dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 380), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 407), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 558), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 677).

**6.555.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 215), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 255), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 407), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 452), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 558), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 677).

**6.555.3.5** `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 256), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 408), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 453), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 678).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h`

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2718).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageMarshaller:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.556.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2718). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.556.2 Constructor & Destructor Documentation

- 6.556.2.1 `activemq::wireformat::openwire::marshal::v6::MessageMarshaller::MessageMarshaller()` [inline]
- 6.556.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

## 6.556.3 Member Function Documentation

- 6.556.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 266), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 467), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 573), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 684).

- 6.556.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

**Exceptions:**

***IOException*** if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 267), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 423), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 468), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 574), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 685).

```
6.556.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat,  commands::DataStructure *
            dataStructure,  utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

***IOException*** if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 267), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 423), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 468), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 574), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 685).



**6.556.3.4** virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 267), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 468), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 574), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 685).

**6.556.3.5** virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 268), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 397), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 424), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 469), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 575), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 686).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h`

## 6.557 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

#include <src/main/cms/MessageNotReadableException.h> Inheritance diagram for cms::MessageNotReadableException:

### Public Member Functions

- **MessageNotReadableException** () throw ()
- **MessageNotReadableException** (const **MessageNotReadableException** &ex) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotReadableException** () throw ()

### 6.557.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since:

1.3

### 6.557.2 Constructor & Destructor Documentation

- 6.557.2.1** cms::MessageNotReadableException::MessageNotReadableException () throw ()
- 6.557.2.2** cms::MessageNotReadableException::MessageNotReadableException (const **MessageNotReadableException** & *ex*) throw ()
- 6.557.2.3** cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message*, const std::exception \* *cause*) throw ()
- 6.557.2.4** cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message*, const std::exception \* *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.557.2.5** virtual  
cms::MessageNotReadableException::~~MessageNotReadableException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageNotReadableException.h

## 6.558 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

#include <src/main/cms/MessageNotWriteableException.h> Inheritance diagram for cms::MessageNotWriteableException:

### Public Member Functions

- **MessageNotWriteableException** () throw ()
- **MessageNotWriteableException** (const **MessageNotWriteableException** &ex) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotWriteableException** () throw ()

### 6.558.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since:

1.3

### 6.558.2 Constructor & Destructor Documentation

- 6.558.2.1** cms::MessageNotWriteableException::MessageNotWriteableException () throw ()
- 6.558.2.2** cms::MessageNotWriteableException::MessageNotWriteableException (const **MessageNotWriteableException** & *ex*) throw ()
- 6.558.2.3** cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message*, const std::exception \* *cause*) throw ()
- 6.558.2.4** cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message*, const std::exception \* *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.558.2.5** virtual  
cms::MessageNotWriteableException::~~MessageNotWriteableException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageNotWriteableException.h

## 6.559 cms::MessageProducer Class Reference

A client uses a `MessageProducer` (p. 2725) object to send messages to a **Destination** (p. 1723).

#include <src/main/cms/MessageProducer.h> Inheritance diagram for cms::MessageProducer:

### Public Member Functions

- virtual `~MessageProducer ()`
- virtual void **send** (**Message** \*message)=0 throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**Message** \*message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **Destination** \*destination, **Message** \*message)=0 throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **Destination** \*destination, **Message** \*message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode)=0 throw ( CMSEException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const =0 throw ( CMSEException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value)=0 throw ( CMSEException )  
*Sets if **Message** (p. 2534) Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const =0 throw ( CMSEException )  
*Gets if **Message** (p. 2534) Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw ( CMSEException )  
*Sets if **Message** (p. 2534) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw ( CMSEException )  
*Gets if **Message** (p. 2534) Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority)=0 throw ( CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const =0 throw ( CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time)=0 throw ( CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const =0 throw ( CMSEException )  
*Gets the Time to Live that this producer sends messages with.*

### 6.559.1 Detailed Description

A client uses a **MessageProducer** (p. 2725) object to send messages to a **Destination** (p. 1723). A **MessageProducer** (p. 2725) object is created by passing a **Destination** (p. 1723) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1723) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since:

1.0

### 6.559.2 Constructor & Destructor Documentation

**6.559.2.1** virtual cms::MessageProducer::~~MessageProducer () [inline, virtual]

### 6.559.3 Member Function Documentation

**6.559.3.1** virtual int cms::MessageProducer::getDeliveryMode () const throw ( CMSEException ) [pure virtual]

Gets the delivery mode for this Producer.

Returns:

The **DeliveryMode** (p. 1721)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1076), and `activemq::core::ActiveMQProducer` (p. 472).

### 6.559.3.2 `virtual bool cms::MessageProducer::getDisableMessageID () const throw ( CMSEException ) [pure virtual]`

Gets if `Message` (p. 2534) Ids are disabled for this Producer.

**Returns:**

boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1077), and `activemq::core::ActiveMQProducer` (p. 472).

### 6.559.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp () const throw ( CMSEException ) [pure virtual]`

Gets if `Message` (p. 2534) Time Stamps are disabled for this Producer.

**Returns:**

boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1077), and `activemq::core::ActiveMQProducer` (p. 472).

### 6.559.3.4 `virtual int cms::MessageProducer::getPriority () const throw ( CMSEException ) [pure virtual]`

Gets the Priority level that this producer sends messages at.

**Returns:**

int based priority level

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1077), and `activemq::core::ActiveMQProducer` (p. 473).

**6.559.3.5** `virtual long long cms::MessageProducer::getTimeToLive () const throw ( CMSExcption )` [pure virtual]

Gets the Time to Live that this producer sends messages with.

**Returns:**

Time to live value in milliseconds

**Exceptions:**

*CMSExcption* (p. 1160) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1078), and `activemq::core::ActiveMQProducer` (p. 473).

**6.559.3.6** `virtual void cms::MessageProducer::send (const Destination * destination, Message * message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSExcption, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )` [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*destination* The destination on which to send the message

*message* The message to be sent.

*deliveryMode* The delivery mode to be used.

*priority* The priority for this message.

*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

*CMSExcption* (p. 1160) - if an internal error occurs while sending the message.

*MessageFormatException* (p. 2662) - if an Invalid **Message** (p. 2534) is given.

*InvalidDestinationException* (p. 2131) - if a client uses this method with a **MessageProducer** (p. 2725) with an invalid destination.

*UnsupportedOperationException* (p. 3919) - if a client uses this method with a **MessageProducer** (p. 2725) that did not specify a destination at creation time.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1078), and `activemq::core::ActiveMQProducer` (p. 474).

**6.559.3.7** `virtual void cms::MessageProducer::send (const Destination * destination, Message * message) throw ( cms::CMSExcption, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )` [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.



**Parameters:**

*destination* The destination on which to send the message  
*message* the message to be sent.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs while sending the message.  
*MessageFormatException* (p. 2662) - if an Invalid **Message** (p. 2534) is given.  
*InvalidDestinationException* (p. 2131) - if a client uses this method with a **MessageProducer** (p. 2725) with an invalid destination.  
*UnsupportedOperationException* (p. 3919) - if a client uses this method with a **MessageProducer** (p. 2725) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1078), and **activemq::core::ActiveMQProducer** (p. 475).

**6.559.3.8** virtual void cms::MessageProducer::send (Message \* *message*,  
int *deliveryMode*, int *priority*, long long *timeToLive*)  
throw ( cms::CMSEException, cms::MessageFormatException,  
cms::InvalidDestinationException, cms::UnsupportedOperationException  
) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

**Parameters:**

*message* The message to be sent.  
*deliveryMode* The delivery mode to be used.  
*priority* The priority for this message.  
*timeToLive* The time to live value for this message in milliseconds.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs while sending the message.  
*MessageFormatException* (p. 2662) - if an Invalid **Message** (p. 2534) is given.  
*InvalidDestinationException* (p. 2131) - if a client uses this method with a **MessageProducer** (p. 2725) with an invalid destination.  
*UnsupportedOperationException* (p. 3919) - if a client uses this method with a **MessageProducer** (p. 2725) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1079), and **activemq::core::ActiveMQProducer** (p. 475).

**6.559.3.9** virtual void cms::MessageProducer::send (Message \* *message*)  
throw ( cms::CMSEException, cms::MessageFormatException,  
cms::InvalidDestinationException, cms::UnsupportedOperationException  
) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

**Parameters:**

*message* The message to be sent.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs while sending the message.

*MessageFormatException* (p. 2662) - if an Invalid **Message** (p. 2534) is given.

*InvalidDestinationException* (p. 2131) - if a client uses this method with a **Message-Producer** (p. 2725) with an invalid destination.

*UnsupportedOperationException* (p. 3919) - if a client uses this method with a **MessageProducer** (p. 2725) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1079), and **activemq::core::ActiveMQProducer** (p. 476).

**6.559.3.10** `virtual void cms::MessageProducer::setDeliveryMode (int mode) throw ( CMSEException )` [pure virtual]

Sets the delivery mode for this Producer.

**Parameters:**

*mode* The **DeliveryMode** (p. 1721)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1080), and **activemq::core::ActiveMQProducer** (p. 476).

**6.559.3.11** `virtual void cms::MessageProducer::setDisableMessageID (bool value) throw ( CMSEException )` [pure virtual]

Sets if **Message** (p. 2534) Ids are disabled for this Producer.

**Parameters:**

*value* boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1080), and **activemq::core::ActiveMQProducer** (p. 476).

**6.559.3.12** `virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool value) throw ( CMSEException )` [pure virtual]

Sets if **Message** (p. 2534) Time Stamps are disabled for this Producer.

**Parameters:**

*value* - boolean indicating enable / disable (true / false)

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1080), and **activemq::core::ActiveMQProducer** (p. 476).

**6.559.3.13 virtual void cms::MessageProducer::setPriority (int *priority*) throw ( CMSEException ) [pure virtual]**

Sets the Priority that this Producers sends messages at.

**Parameters:**

*priority* int value for Priority level

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1081), and **activemq::core::ActiveMQProducer** (p. 477).

**6.559.3.14 virtual void cms::MessageProducer::setTimeToLive (long long *time*) throw ( CMSEException ) [pure virtual]**

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

**Parameters:**

*time* default time to live value in milliseconds

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1081), and **activemq::core::ActiveMQProducer** (p. 477).

The documentation for this class was generated from the following file:

- src/main/cms/MessageProducer.h

## 6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
```

### Public Member Functions

- **MessagePropertyInterceptor** (**commands::Message** \*message, **util::PrimitiveMap** \*properties) throw ( **decaf::lang::exceptions::NullPointerException** )  
*Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.*
- virtual **~MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const  
*Gets a float property.*
- virtual int **getIntProperty** (const std::string &name) const  
*Gets a int property.*
- virtual long long **getLongProperty** (const std::string &name) const  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const  
*Gets a string property.*
- virtual void **setBooleanProperty** (const std::string &name, bool value)  
*Sets a boolean property.*
- virtual void **setByteProperty** (const std::string &name, unsigned char value)  
*Sets a byte property.*
- virtual void **setDoubleProperty** (const std::string &name, double value)  
*Sets a double property.*

- virtual void **setFloatProperty** (const std::string &name, float value)  
*Sets a float property.*
- virtual void **setIntProperty** (const std::string &name, int value)  
*Sets a int property.*
- virtual void **setLongProperty** (const std::string &name, long long value)  
*Sets a long property.*
- virtual void **setShortProperty** (const std::string &name, short value)  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value)  
*Sets a string property.*

## 6.560.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name		Conversion Supported	-----	JMSXDeliveryCount	
Int, Long, String		JMSXGroupID		String	
		JMSXGroupSeq		Int, Long, String	

## 6.560.2 Constructor & Destructor Documentation

### 6.560.2.1 activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor (commands::Message \* *message*, util::PrimitiveMap \* *properties*) throw ( decaf::lang::exceptions::NullPointerException )

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

#### Parameters:

- message* - The Message to store reserved property data in
- properties* - The PrimitiveMap to store the rest of the properties in.

#### Exceptions:

- NullPointerException* if either param is NULL

**6.560.2.2**    **virtual**  
          **activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~~MessagePropertyInt**  
          **()**    [virtual]

### 6.560.3    Member Function Documentation

**6.560.3.1**    **virtual bool ac-**  
          **tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty**  
          **(const std::string & name) const**    [virtual]

Gets a boolean property.

**Parameters:**

*name*    The name of the property to retrieve.

**Returns:**

      The value for the named property.

**6.560.3.2**    **virtual unsigned char ac-**  
          **tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBytesProperty**  
          **(const std::string & name) const**    [virtual]

Gets a byte property.

**Parameters:**

*name*    The name of the property to retrieve.

**Returns:**

      The value for the named property.

**6.560.3.3**    **virtual double ac-**  
          **tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty**  
          **(const std::string & name) const**    [virtual]

Gets a double property.

**Parameters:**

*name*    The name of the property to retrieve.

**Returns:**

      The value for the named property.

**6.560.3.4**    **virtual float ac-**  
          **tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty**  
          **(const std::string & name) const**    [virtual]

Gets a float property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**6.560.3.5** `virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty(const std::string & name) const [virtual]`

Gets a int property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**6.560.3.6** `virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty(const std::string & name) const [virtual]`

Gets a long property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**6.560.3.7** `virtual short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty(const std::string & name) const [virtual]`

Gets a short property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**6.560.3.8** `virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty(const std::string & name) const [virtual]`

Gets a string property.

**Parameters:**

*name* The name of the property to retrieve.

**Returns:**

The value for the named property.

**6.560.3.9** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty(const std::string & name, bool value) [virtual]`

Sets a boolean property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.10** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty(const std::string & name, unsigned char value) [virtual]`

Sets a byte property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.11** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty(const std::string & name, double value) [virtual]`

Sets a double property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.



**6.560.3.12** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty(const std::string & *name*, float *value*) [virtual]

Sets a float property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.13** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty(const std::string & *name*, int *value*) [virtual]

Sets a int property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.14** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty(const std::string & *name*, long long *value*) [virtual]

Sets a long property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.15** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty(const std::string & *name*, short *value*) [virtual]

Sets a short property.

**Parameters:**

*name* The name of the property to retrieve.

*value* The value for the named property.

**6.560.3.16** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty(const std::string & *name*, const std::string & *value*) [virtual]

Sets a string property.

**Parameters:**

***name*** The name of the property to retrieve.

***value*** The value for the named property.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h`

## 6.561 activemq::commands::MessagePull Class Reference

#include <src/main/activemq/commands/MessagePull.h> Inheritance diagram for activemq::commands::MessagePull:

### Public Member Functions

- **MessagePull** ()
- virtual **~MessagePull** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessagePull \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEPULL** = 20

## Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

## 6.561.1 Constructor & Destructor Documentation

**6.561.1.1** **activemq::commands::MessagePull::MessagePull ()**

**6.561.1.2** **virtual activemq::commands::MessagePull::~~MessagePull ()** [virtual]

## 6.561.2 Member Function Documentation

**6.561.2.1** **virtual MessagePull\* activemq::commands::MessagePull::cloneDataStructure ()**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.561.2.2** **virtual void activemq::commands::MessagePull::copyDataStructure**  
**(const DataStructure \* src)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.561.2.3** **virtual bool activemq::commands::MessagePull::equals (const**  
**DataStructure \* value)** **const** [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

- 6.561.2.4** virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ()  
[virtual]
- 6.561.2.5** virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () const  
[virtual]
- 6.561.2.6** virtual std::string& activemq::commands::MessagePull::getCorrelationId ()  
[virtual]
- 6.561.2.7** virtual const std::string& activemq::commands::MessagePull::getCorrelationId () const  
[virtual]
- 6.561.2.8** virtual unsigned char activemq::commands::MessagePull::getDataStructureType () const  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.561.2.9    virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination ()  
[virtual]
- 6.561.2.10   virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const [virtual]
- 6.561.2.11   virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId ()  
[virtual]
- 6.561.2.12   virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const  
[virtual]
- 6.561.2.13   virtual long long activemq::commands::MessagePull::getTimeout ()  
const [virtual]
- 6.561.2.14   virtual void activemq::commands::MessagePull::setConsumerId (const  
Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.561.2.15   virtual void activemq::commands::MessagePull::setCorrelationId (const  
std::string & *correlationId*) [virtual]
- 6.561.2.16   virtual void activemq::commands::MessagePull::setDestination (const  
Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.561.2.17   virtual void activemq::commands::MessagePull::setMessageId (const  
Pointer< MessageId > & *messageId*) [virtual]
- 6.561.2.18   virtual void activemq::commands::MessagePull::setTimeout (long long  
*timeout*) [virtual]
- 6.561.2.19   virtual std::string activemq::commands::MessagePull::toString () const  
[virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.561.2.20   virtual Pointer<Command> activemq::commands::MessagePull::visit  
(activemq::state::CommandVisitor \* *visitor*) throw (  
exceptions::ActiveMQException ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

**6.561.3 Field Documentation**

**6.561.3.1** **Pointer<ConsumerId> activemq::commands::MessagePull::consumerId**  
[protected]

**6.561.3.2** **std::string activemq::commands::MessagePull::correlationId** [protected]

**6.561.3.3** **Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination** [protected]

**6.561.3.4** **const unsigned char activemq::commands::MessagePull::ID \_ - MESSAGEPULL = 20** [static]

**6.561.3.5** **Pointer<MessageId> activemq::commands::MessagePull::messageId**  
[protected]

**6.561.3.6** **long long activemq::commands::MessagePull::timeout** [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

## 6.562 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2744).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.562.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2744).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.562.2 Constructor & Destructor Documentation

**6.562.2.1** `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.562.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.562.3 Member Function Documentation

**6.562.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.562.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.562.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.562.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.562.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.562.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.562.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

## 6.563 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2748).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.563.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2748).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.563.2 Constructor & Destructor Documentation

**6.563.2.1** `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.563.2.2** `virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.563.3 Member Function Documentation

**6.563.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.563.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.563.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.563.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.563.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.563.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.563.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h

## 6.564 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2752).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.564.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2752).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.564.2 Constructor & Destructor Documentation

**6.564.2.1** `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.564.2.2** `virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.564.3 Member Function Documentation

**6.564.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.564.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.564.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.564.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.564.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.564.3.6** virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.564.3.7** virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h

## 6.565 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2756).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.565.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2756).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.565.2 Constructor & Destructor Documentation

**6.565.2.1** `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.565.2.2** `virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.565.3 Member Function Documentation

**6.565.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.565.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.565.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.565.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.565.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.565.3.6** virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.565.3.7** virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

## 6.566 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2760).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.566.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2760).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.566.2 Constructor & Destructor Documentation

**6.566.2.1** `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.566.2.2** `virtual activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.566.3 Member Function Documentation

**6.566.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.566.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.566.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.566.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.566.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.566.3.6** virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.566.3.7** virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

## 6.567 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2764).

#include <src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.567.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **MessagePullMarshaller** (p.2764).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.567.2 Constructor & Destructor Documentation

**6.567.2.1** `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::MessagePullMarshaller()` [inline]

**6.567.2.2** `virtual activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

## 6.567.3 Member Function Documentation

**6.567.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.567.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.567.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.567.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.567.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.567.3.6** virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.567.3.7** virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h

## 6.568 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 2768) defines a base level **Transport** (p. 3883) class that is intended to be used in place of an a regular protocol **Transport** (p. 3883) such as TCP.

#include <src/main/activemq/transport/mock/MockTransport.h> Inheritance diagram for activemq::transport::mock::MockTransport:

### Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual ~**MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)  
*Sets the **ResponseBuilder** (p. 3289) that this class uses to create Responses to Commands sent.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)  
*Sends the given command to the broker and then waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)  
*Sends the given command to the broker and then waits for the response.*
- virtual void **setOutgoingListener** (**TransportListener** \*listener)  
*Sets a Listener that gets notified for every command that would have been sent by this **transport** (p. 97) to the Broker, this allows a client to verify that its messages are making it to the wire.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat **AMQCPP\_UNUSED**)  
*Sets the WireFormat instance to use.*
- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () const  
*Gets the currently set WireFormat.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual void **fireCommand** (const **Pointer**< **Command** > &command)



*Fires a Command back through this **transport** (p. 97) to its registered CommandListener if there is one.*

- virtual void **fireException** (const exceptions::ActiveMQException &ex)  
*Fires a Exception back through this **transport** (p. 97) to its registered ExceptionListener if there is one.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts the **Transport** (p. 3883), the send methods of a **Transport** (p. 3883) will throw an exception if used before the **Transport** (p. 3883) is started.*
- virtual void **stop** () throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3883).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeId)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3883) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3883) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*reconnect to another location*
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSentMessageBeforeFail** () const
- void **setNumSentMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const

- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

## Static Public Member Functions

- static **MockTransport** \* **getInstance** ()

### 6.568.1 Detailed Description

The **MockTransport** (p. 2768) defines a base level **Transport** (p. 3883) class that is intended to be used in place of an a regular protocol **Transport** (p. 3883) such as TCP. This **Transport** (p. 3883) assumes that it is the base **Transport** (p. 3883) in the Transports stack, and destroys any **Transports** that are passed to it in its constructor.

This **Transport** (p. 3883) defines an Interface **ResponseBuilder** (p. 3289) which must be implemented by any protocol for which the **Transport** (p. 3883) is used to Emulate. The **Transport** (p. 3883) hands off all outbound **commands** (p. 87) to the **ResponseBuilder** (p. 3289) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

### 6.568.2 Constructor & Destructor Documentation

- 6.568.2.1** **activemq::transport::mock::MockTransport::MockTransport** (const **Pointer**< **wireformat::WireFormat** > & *wireFormat*, const **Pointer**< **ResponseBuilder** > & *responseBuilder*)
- 6.568.2.2** **virtual activemq::transport::mock::MockTransport::~MockTransport** ()  
[inline, virtual]

### 6.568.3 Member Function Documentation

- 6.568.3.1** **virtual void activemq::transport::mock::MockTransport::close** () **throw** ( **decaf::io::IOException** ) [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

#### Exceptions:

***IOException*** if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 1149).

**6.568.3.2 virtual void activemq::transport::mock::MockTransport::fireCommand (const Pointer< Command > & *command*) [inline, virtual]**

Fires a Command back through this **transport** (p. 97) to its registered CommandListener if there is one.

**Parameters:**

*command* - Command to send to the Listener.

**6.568.3.3 virtual void activemq::transport::mock::MockTransport::fireException (const exceptions::ActiveMQException & *ex*) [inline, virtual]**

Fires a Exception back through this **transport** (p. 97) to its registered ExceptionListener if there is one.

**Parameters:**

*ex* The Exception that will be passed on the the **Transport** (p. 3883) listener.

**6.568.3.4 static MockTransport\* activemq::transport::mock::MockTransport::getInstance () [inline, static]****6.568.3.5 int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail () const [inline]****6.568.3.6 int activemq::transport::mock::MockTransport::getNumReceivedMessages () const [inline]****6.568.3.7 int activemq::transport::mock::MockTransport::getNumSentKeepAlives () const [inline]****6.568.3.8 int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail () const [inline]****6.568.3.9 int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail () const [inline]****6.568.3.10 int activemq::transport::mock::MockTransport::getNumSentMessages () const [inline]****6.568.3.11 virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress () const [inline, virtual]****Returns:**

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3884).

**6.568.3.12** `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).

**Returns:**

The listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3884).

**6.568.3.13** `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat () const [inline]`

Gets the currently set WireFormat.

**Returns:**

the current WireFormat object.

**6.568.3.14** `virtual bool activemq::transport::mock::MockTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3883) been shutdown and no longer usable.

**Returns:**

true if the **Transport** (p. 3883)

Implements **activemq::transport::Transport** (p. 3885).

**6.568.3.15** `virtual bool activemq::transport::mock::MockTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3883) Connected to its Broker.

**Returns:**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3885).

- 6.568.3.16 `bool activemq::transport::mock::MockTransport::isFailOnClose () const [inline]`
- 6.568.3.17 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends () const [inline]`
- 6.568.3.18 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage () const [inline]`
- 6.568.3.19 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage () const [inline]`
- 6.568.3.20 `bool activemq::transport::mock::MockTransport::isFailOnStart () const [inline]`
- 6.568.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStop () const [inline]`
- 6.568.3.22 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns:

true if the **Transport** (p. 3883) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3885).

- 6.568.3.23 `virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.

#### Parameters:

*typeId* - The `type_info` of the Object we are searching for.

#### Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3885).

- 6.568.3.24 `virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

**Parameters:**

*command* the command to be sent.

**Exceptions:**

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3886).

**6.568.3.25** virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri *AMQCPP\_UNUSED*) throw ( decaf::io::IOException ) [inline, virtual]

reconnect to another location

**Parameters:**

*uri*

**Exceptions:**

*IOException* on failure of if not supported

**6.568.3.26** virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* - The command to be sent.

*timeout* - The time to wait for this response.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3886).

**6.568.3.27** virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* the command to be sent.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3887).

- 6.568.3.28 `void activemq::transport::mock::MockTransport::setFailOnClose (bool value) [inline]`
- 6.568.3.29 `void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends (bool value) [inline]`
- 6.568.3.30 `void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool value) [inline]`
- 6.568.3.31 `void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool value) [inline]`
- 6.568.3.32 `void activemq::transport::mock::MockTransport::setFailOnStart (bool value) [inline]`
- 6.568.3.33 `void activemq::transport::mock::MockTransport::setFailOnStop (bool value) [inline]`
- 6.568.3.34 `void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int value) [inline]`
- 6.568.3.35 `void activemq::transport::mock::MockTransport::setNumReceivedMessages (int value) [inline]`
- 6.568.3.36 `void activemq::transport::mock::MockTransport::setNumSentKeepAlives (int value) [inline]`
- 6.568.3.37 `void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail (int value) [inline]`
- 6.568.3.38 `void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int value) [inline]`
- 6.568.3.39 `void activemq::transport::mock::MockTransport::setNumSentMessages (int value) [inline]`
- 6.568.3.40 `virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * listener) [inline, virtual]`

Sets a Listener that gets notified for every command that would have been sent by this **transport** (p. 97) to the Broker, this allows a client to verify that its messages are making it to the wire.

#### Parameters:

*listener* - The CommandListener to notify for each message



**6.568.3.41 void activemq::transport::mock::MockTransport::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]**

Sets the **ResponseBuilder** (p. 3289) that this class uses to create Responses to Commands sent. These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

**Parameters:**

*responseBuilder* - The **ResponseBuilder** (p. 3289) to use from now on.

**6.568.3.42 virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener \* *listener*) [inline, virtual]**

Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).

**Parameters:**

*listener* the listener of **transport** (p. 97) events.

Implements **activemq::transport::Transport** (p. 3887).

**6.568.3.43 virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP\_UNUSED*) [inline, virtual]**

Sets the WireFormat instance to use.

**Parameters:**

*wireFormat* WireFormat the object used to encode / decode **commands** (p. 87).

**6.568.3.44 virtual void activemq::transport::mock::MockTransport::start () throw ( decaf::io::IOException ) [virtual]**

Starts the **Transport** (p. 3883), the send methods of a **Transport** (p. 3883) will throw an exception if used before the **Transport** (p. 3883) is started.

**Exceptions:**

*IOException* if and error occurs while starting the **Transport** (p. 3883).

Implements **activemq::transport::Transport** (p. 3888).

**6.568.3.45 virtual void activemq::transport::mock::MockTransport::stop () throw ( decaf::io::IOException ) [virtual]**

Stops the **Transport** (p. 3883).

**Exceptions:**

*IOException* if an error occurs while stopping the **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3888).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransport.h`

## 6.569 activemq::transport::mock::MockTransportFactory Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

#include <src/main/activemq/transport/mock/MockTransportFactory.h>Inheritance diagram for activemq::transport::mock::MockTransportFactory:

### Public Member Functions

- virtual `~MockTransportFactory ()`
- virtual `Pointer< Transport > create (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`

*Creates a fully configured **Transport** (p. 3883) instance which could be a chain of filters and transports.*

- virtual `Pointer< Transport > createComposite (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`

*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

### Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw ( exceptions::ActiveMQException )`

*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

#### 6.569.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

## 6.569.2 Constructor & Destructor Documentation

**6.569.2.1** `virtual  
activemq::transport::mock::MockTransportFactory::~MockTransportFactory  
( ) [inline, virtual]`

## 6.569.3 Member Function Documentation

**6.569.3.1** `virtual Pointer<Transport> ac-  
tivismq::transport::mock::MockTransportFactory::create (const  
decaf::net::URI & location) throw ( exceptions::ActiveMQException )  
[virtual]`

Creates a fully configured **Transport** (p. 3883) instance which could be a chain of filters and transports.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements `activemq::transport::TransportFactory` (p. 3889).

**6.569.3.2** `virtual Pointer<Transport> ac-  
tivismq::transport::mock::MockTransportFactory::createComposite (const  
decaf::net::URI & location) throw ( exceptions::ActiveMQException )  
[virtual]`

Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements `activemq::transport::TransportFactory` (p. 3890).

**6.569.3.3** `virtual Pointer<Transport> ac-  
tivismq::transport::mock::MockTransportFactory::doCreateComposite  
(const decaf::net::URI & location, const Pointer<  
wireformat::WireFormat > & wireFormat, const decaf::util::Properties  
& properties) throw ( exceptions::ActiveMQException ) [protected,  
virtual]`

Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.

**Parameters:**

*location* - URI location to connect to.

*wireFormat* - the assigned WireFormat for the new **Transport** (p. 3883).

*properties* - Properties to apply to the **transport** (p. 97).

**Returns:**

Pointer to a new **Transport** (p. 3883) instance.

**Exceptions:**

*ActiveMQException* if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

## 6.570 decaf::util::concurrent::Mutex Class Reference

**Mutex** (p.2782) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

#include <src/main/decaf/util/concurrent/Mutex.h> Inheritance diagram for decaf::util::concurrent::Mutex:

### Public Member Functions

- **Mutex** ()
- virtual ~**Mutex** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to **Lock** (p.2375) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

## 6.570.1 Detailed Description

**Mutex** (p.2782) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

**Since:**

1.0

## 6.570.2 Constructor & Destructor Documentation

**6.570.2.1** decaf::util::concurrent::Mutex::Mutex ()

**6.570.2.2** virtual decaf::util::concurrent::Mutex::~~Mutex () [virtual]

## 6.570.3 Member Function Documentation

**6.570.3.1** virtual void decaf::util::concurrent::Mutex::lock () throw ( decaf::lang::exceptions::RuntimeException ) [virtual]

Locks the object.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p.3701).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock(), decaf::util::StlMap< std::string, cms::Topic \* >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::AbstractCollection< cms::Connection \* >::lock().

**6.570.3.2** virtual void decaf::util::concurrent::Mutex::notify () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p.3700) Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p.3702).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify(), decaf::util::StlMap< std::string, cms::Topic \* >::notify(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify(), and decaf::util::AbstractCollection< cms::Connection \* >::notify().

**6.570.3.3** `virtual void decaf::util::concurrent::Mutex::notifyAll  
() throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

***RuntimeException*** if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll()`, `decaf::util::StlMap< std::string, cms::Topic * >::notifyAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`.

**6.570.3.4** `virtual bool decaf::util::concurrent::Mutex::tryLock () throw (  
decaf::lang::exceptions::RuntimeException ) [virtual]`

Attempts to **Lock** (p. 2375) the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

***RuntimeException*** if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`.

**6.570.3.5** `virtual void decaf::util::concurrent::Mutex::unlock () throw (  
decaf::lang::exceptions::RuntimeException ) [virtual]`

Unlocks the object.

**Exceptions:**

***RuntimeException*** if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).



Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::unlock()`.

**6.570.3.6** `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters:

*millisecs* the time in milliseconds to wait, or `WAIT_INFINITE`

*nanos* additional time in nanoseconds with a range of 0-999999

#### Exceptions:

***IllegalArgumentException*** if an error occurs or the nanos argument is not in the range of [0-999999]

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.570.3.7** `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters:

*millisecs* the time in milliseconds to wait, or `WAIT_INFINITE`

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

**6.570.3.8** **virtual void decaf::util::concurrent::Mutex::wait ()**  
**throw ( decaf::lang::exceptions::RuntimeException,**  
**decaf::lang::exceptions::IllegalMonitorStateException,**  
**decaf::lang::exceptions::InterruptedException ) [virtual]**

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Mutex.h`

## 6.571 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

### Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

### Data Fields

- pthread\_mutex\_t **mutex**
- volatile long long **lock\_owner**
- volatile long long **lock\_count**
- CRITICAL\_SECTION **mutex**

### 6.571.1 Constructor & Destructor Documentation

**6.571.1.1** decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

**6.571.1.2** decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

**6.571.1.3** decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

**6.571.1.4** decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

### 6.571.2 Field Documentation

**6.571.2.1** volatile long long decaf::util::concurrent::MutexHandle::lock\_count

**6.571.2.2** volatile long long decaf::util::concurrent::MutexHandle::lock\_owner

**6.571.2.3** CRITICAL\_SECTION decaf::util::concurrent::MutexHandle::mutex

**6.571.2.4** pthread\_mutex\_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**MutexHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**MutexHandle.h**

## 6.572 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

### Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle \* create** ()  
*Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.*
- static void **destroy** (decaf::util::concurrent::MutexHandle \*handle)  
*Destroy a previously create Mutex instance.*
- static void **lock** (decaf::util::concurrent::MutexHandle \*handle)  
*Locks the Mutex.*
- static bool **trylock** (decaf::util::concurrent::MutexHandle \*handle)  
*Tries to lock the Mutex.*
- static void **unlock** (decaf::util::concurrent::MutexHandle \*handle)  
*Unlocks the Mutex allowing other Thread to then acquire the Lock on it.*

### 6.572.1 Member Function Documentation

#### 6.572.1.1 static decaf::util::concurrent::MutexHandle\* decaf::internal::util::concurrent::MutexImpl::create () [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

#### Returns:

handle to a newly created Mutex.

#### 6.572.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (decaf::util::concurrent::MutexHandle \* handle) [static]

Destroy a previously create Mutex instance.

#### Parameters:

*mutex* The Mutex instance to be destroyed.

**6.572.1.3 static void decaf::internal::util::concurrent::MutexImpl::lock**  
(decaf::util::concurrent::MutexHandle \* *handle*) [static]

Locks the Mutex. If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

**Parameters:**

*handle* the handle to the Mutex to Lock.

**6.572.1.4 static bool decaf::internal::util::concurrent::MutexImpl::trylock**  
(decaf::util::concurrent::MutexHandle \* *handle*) [static]

Tries to lock the Mutex. If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

**Parameters:**

*handle* the handle to the Mutex to attempt to Lock.

**Returns:**

true if the lock was acquired false otherwise.

**6.572.1.5 static void decaf::internal::util::concurrent::MutexImpl::unlock**  
(decaf::util::concurrent::MutexHandle \* *handle*) [static]

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

**Parameters:**

*handle* the handle to the Mutex to attempt to Lock.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/MutexImpl.h

## 6.573 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

```
#include <src/main/decaf/internal/net/Network.h>
```

### Public Member Functions

- virtual `~Network ()`
- `decaf::util::concurrent::Mutex * getRuntimeLock ()`  
*Gets a pointer to the **Network** (p. 2790) Runtime's Lock object, this can be used by **Network** (p. 2790) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2790) layer, etc.*
- void `addNetworkResource (decaf::internal::util::Resource *value)`  
*Adds a Resource to the **Network** (p. 2790) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2790) Runtime are destroyed.*
- template<typename T >  
void `addAsResource (T *value)`

### Static Public Member Functions

- static `Network * getNetworkRuntime ()`  
*Gets the one and only instance of the **Network** (p. 2790) class, if this is called before the **Network** (p. 2790) layer has been initialized or after it has been shutdown then an *IllegalStateException* is thrown.*
- static void `initializeNetworking ()`  
*Initialize the Networking layer.*
- static void `shutdownNetworking ()`  
*Shutdown the **Network** (p. 2790) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

### Protected Member Functions

- `Network ()`

#### 6.573.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

Since:

1.0

## 6.573.2 Constructor & Destructor Documentation

**6.573.2.1** `decaf::internal::net::Network::Network ()` [protected]

**6.573.2.2** `virtual decaf::internal::net::Network::~~Network ()` [virtual]

## 6.573.3 Member Function Documentation

**6.573.3.1** `template<typename T > void decaf::internal::net::Network::addAsResource (T * value)`  
[inline]

**6.573.3.2** `void decaf::internal::net::Network::addNetworkResource (decaf::internal::util::Resource * value)`

Adds a Resource to the **Network** (p. 2790) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2790) Runtime are destroyed.

### Parameters:

*value* The Resource to add to the **Network** (p. 2790) Runtime.

### Exceptions:

*NullPointerException* if the Resource value passed is null.

**6.573.3.3** `static Network* decaf::internal::net::Network::getNetworkRuntime ()`  
[static]

Gets the one and only instance of the **Network** (p. 2790) class, if this is called before the **Network** (p. 2790) layer has been initialized or after it has been shutdown then an *IllegalStateException* is thrown.

### Returns:

pointer to the **Network** (p. 2790) runtime for the Decaf library.

**6.573.3.4** `decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ()`

Gets a pointer to the **Network** (p. 2790) Runtime's Lock object, this can be used by **Network** (p. 2790) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2790) layer, etc. The pointer returned is owned by the **Network** (p. 2790) runtime and should not be deleted or copied by the caller.

### Returns:

a pointer to the **Network** (p. 2790) Runtime's single Lock instance.

**6.573.3.5** `static void decaf::internal::net::Network::initializeNetworking ()` [static]

Initialize the Networking layer.

### 6.573.3.6 `static void decaf::internal::net::Network::shutdownNetworking ()` [static]

Shutdown the **Network** (p. 2790) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/Network.h`



## 6.574 activemq::commands::NetworkBridgeFilter Class Reference

#include <src/main/activemq/commands/NetworkBridgeFilter.h> Inheritance diagram for activemq::commands::NetworkBridgeFilter:

### Public Member Functions

- **NetworkBridgeFilter** ()
- virtual **~NetworkBridgeFilter** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **NetworkBridgeFilter \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int networkTTL)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &networkBrokerId)

### Static Public Attributes

- static const unsigned char **ID\_NETWORKBRIDGEFILTER** = 91

### Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

## 6.574.1 Constructor & Destructor Documentation

**6.574.1.1** `activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter ()`

**6.574.1.2** `virtual  
activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter ()  
[virtual]`

## 6.574.2 Member Function Documentation

**6.574.2.1** `virtual NetworkBridgeFilter* ac-  
tivismq::commands::NetworkBridgeFilter::cloneDataStructure () const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.574.2.2** `virtual void ac-  
tivismq::commands::NetworkBridgeFilter::copyDataStructure (const  
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.574.2.3** `virtual bool activismq::commands::NetworkBridgeFilter::equals (const  
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

**6.574.2.4** `virtual unsigned char ac-  
tivismq::commands::NetworkBridgeFilter::getDataStructureType () const  
[virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.574.2.5 `virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId ()` [virtual]
- 6.574.2.6 `virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const` [virtual]
- 6.574.2.7 `virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const` [virtual]
- 6.574.2.8 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & networkBrokerId)` [virtual]
- 6.574.2.9 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int networkTTL)` [virtual]
- 6.574.2.10 `virtual std::string activemq::commands::NetworkBridgeFilter::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.833).

### 6.574.3 Field Documentation

- 6.574.3.1 `const unsigned char activemq::commands::NetworkBridgeFilter::ID _ - NETWORKBRIDGEFILTER = 91` [static]
- 6.574.3.2 `Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId` [protected]
- 6.574.3.3 `int activemq::commands::NetworkBridgeFilter::networkTTL` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/NetworkBridgeFilter.h`

## 6.575 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2796).

#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.575.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2796). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.575.2 Constructor & Destructor Documentation

**6.575.2.1** `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

**6.575.2.2** `virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

## 6.575.3 Member Function Documentation

**6.575.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.575.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.575.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.575.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.575.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.575.3.6** virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.575.3.7** virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h

## 6.576 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2800).

#include <src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.576.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2800). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.576.2 Constructor & Destructor Documentation

**6.576.2.1** `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

**6.576.2.2** `virtual activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

## 6.576.3 Member Function Documentation

**6.576.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.576.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.576.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.576.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.576.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.576.3.6** virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.576.3.7** virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h

## 6.577 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2804).

#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.577.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2804). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.577.2 Constructor & Destructor Documentation

**6.577.2.1** `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

**6.577.2.2** `virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

## 6.577.3 Member Function Documentation

**6.577.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.577.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.577.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.577.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.577.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.577.3.6** virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.577.3.7** virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h

## 6.578 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2808).

#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.578.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2808). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.578.2 Constructor & Destructor Documentation

**6.578.2.1** `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

**6.578.2.2** `virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

## 6.578.3 Member Function Documentation

**6.578.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.578.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.578.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.578.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.578.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.578.3.6** virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.578.3.7** virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h

## 6.579 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2812).

#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.579.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2812). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.579.2 Constructor & Destructor Documentation

**6.579.2.1** `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

**6.579.2.2** `virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

## 6.579.3 Member Function Documentation

**6.579.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.579.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.579.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.579.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.579.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.579.3.6** virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.579.3.7** virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h

## 6.580 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2816).

#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.580.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2816). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.580.2 Constructor & Destructor Documentation

**6.580.2.1** `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` `[inline]`

**6.580.2.2** `virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` `[inline, virtual]`

## 6.580.3 Member Function Documentation

**6.580.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.580.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.580.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.580.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.580.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.580.3.6** virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.580.3.7** virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h

## 6.581 decaf::net::NoRouteToHostException Class Reference

#include <src/main/decaf/net/NoRouteToHostException.h> Inheritance diagram for decaf::net::NoRouteToHostException:

### Public Member Functions

- **NoRouteToHostException** () throw ()  
*Default Constructor.*
- **NoRouteToHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()  
*Copy Constructor.*
- **NoRouteToHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoRouteToHostException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoRouteToHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoRouteToHostException** \* clone () const  
*Clones this exception.*
- virtual ~**NoRouteToHostException** () throw ()

### 6.581.1 Constructor & Destructor Documentation

**6.581.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException () throw () [inline]**

Default Constructor.

**6.581.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & ex) throw () [inline]**

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.581.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.581.1.4 decaf::net::NoRouteToHostException::NoRouteToHostException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.581.1.5 decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.581.1.6 decaf::net::NoRouteToHostException::NoRouteToHostException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.581.1.7**    **virtual**  
**decaf::net::NoRouteToHostException::~~NoRouteToHostException ()**  
**throw ()**    [inline, virtual]

## **6.581.2    Member Function Documentation**

**6.581.2.1**    **virtual NoRouteToHostException\* de-**  
**caf::net::NoRouteToHostException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3522).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/NoRouteToHostException.h`

## 6.582 decaf::security::NoSuchAlgorithmException Class Reference

#include <src/main/decaf/security/NoSuchAlgorithmException.h> Inheritance diagram for decaf::security::NoSuchAlgorithmException:

### Public Member Functions

- **NoSuchAlgorithmException** () throw ()  
*Default Constructor.*
- **NoSuchAlgorithmException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchAlgorithmException** (const **NoSuchAlgorithmException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchAlgorithmException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchAlgorithmException** \* clone () const  
*Clones this exception.*
- virtual ~**NoSuchAlgorithmException** () throw ()

### 6.582.1 Constructor & Destructor Documentation

#### 6.582.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]

Default Constructor.

#### 6.582.1.2 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

**6.582.1.3** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`  
(const NoSuchAlgorithmException & *ex*) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

**6.582.1.4** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`  
(const char \* *file*, const int *lineNumber*, const std::exception \* *cause*,  
const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.582.1.5** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`  
(const std::exception \* *cause*) throw () [inline]

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.582.1.6** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`  
(const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message



**6.582.1.7**    **virtual**  
**decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException**  
**() throw ()**    [inline, virtual]

## 6.582.2 Member Function Documentation

**6.582.2.1**    **virtual** **NoSuchAlgorithmException\*** **de-**  
**caf::security::NoSuchAlgorithmException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1973).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

## 6.583 decaf::lang::exceptions::NoSuchElementException Class Reference

#include <src/main/decaf/lang/exceptions/NoSuchElementException.h> Inheritance diagram for decaf::lang::exceptions::NoSuchElementException:

### Public Member Functions

- **NoSuchElementException** () throw ()  
*Default Constructor.*
- **NoSuchElementException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchElementException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchElementException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchElementException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchElementException** \* clone () const  
*Clones this exception.*
- virtual ~**NoSuchElementException** () throw ()

### 6.583.1 Constructor & Destructor Documentation

#### 6.583.1.1 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException () throw () [inline]

Default Constructor.

#### 6.583.1.2 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.583.1.3 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**  
(const NoSuchElementException & *ex*) throw () [inline]

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.583.1.4 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**  
(const char \* *file*, const int *lineNumber*, const std::exception \* *cause*,  
const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.583.1.5 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**  
(const std::exception \* *cause*) throw () [inline]

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.583.1.6 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**  
(const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.583.1.7**   **virtual**  
**decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException**  
**() throw ()**   [inline, virtual]

## **6.583.2   Member Function Documentation**

**6.583.2.1**   **virtual** **NoSuchElementException\*** **de-**  
**caf::lang::exceptions::NoSuchElementException::clone ()**  
**const**   [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NoSuchElementException.h`

## 6.584 decaf::security::NoSuchProviderException Class Reference

#include <src/main/decaf/security/NoSuchProviderException.h> Inheritance diagram for decaf::security::NoSuchProviderException:

### Public Member Functions

- **NoSuchProviderException** () throw ()  
*Default Constructor.*
- **NoSuchProviderException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchProviderException** (const **NoSuchProviderException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchProviderException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchProviderException** \* clone () const  
*Clones this exception.*
- virtual ~**NoSuchProviderException** () throw ()

### 6.584.1 Constructor & Destructor Documentation

#### 6.584.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException () throw () [inline]

Default Constructor.

#### 6.584.1.2 decaf::security::NoSuchProviderException::NoSuchProviderException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.584.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.584.1.4 decaf::security::NoSuchProviderException::NoSuchProviderException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.584.1.5 decaf::security::NoSuchProviderException::NoSuchProviderException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.584.1.6 decaf::security::NoSuchProviderException::NoSuchProviderException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.584.1.7**    **virtual**  
**decaf::security::NoSuchProviderException::~~NoSuchProviderException**  
**() throw ()**    [inline, virtual]

## 6.584.2 Member Function Documentation

**6.584.2.1**    **virtual** **NoSuchProviderException\*** **de-**  
**caf::security::NoSuchProviderException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1973).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

## 6.585 decaf::lang::exceptions::NullPointerException Class Reference

#include <src/main/decaf/lang/exceptions/NullPointerException.h> Inheritance diagram for decaf::lang::exceptions::NullPointerException:

### Public Member Functions

- **NullPointerException** () throw ()  
*Default Constructor.*
- **NullPointerException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()  
*Copy Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NullPointerException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NullPointerException** \* clone () const  
*Clones this exception.*
- virtual ~**NullPointerException** () throw ()

### 6.585.1 Constructor & Destructor Documentation

#### 6.585.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

#### 6.585.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.



**6.585.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException  
(const NullPointerException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p.1831) whose data is to be copied into this one.

**6.585.1.4 decaf::lang::exceptions::NullPointerException::NullPointerException  
(const char \* *file*, const int *lineNumber*, const std::exception \* *cause*,  
const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.585.1.5 decaf::lang::exceptions::NullPointerException::NullPointerException  
(const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p.2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.585.1.6 decaf::lang::exceptions::NullPointerException::NullPointerException  
(const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()  
[inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.585.1.7**    **virtual**  
**decaf::lang::exceptions::NullPointerException::~~NullPointerException ()**  
**throw ()**    [inline, virtual]

## **6.585.2    Member Function Documentation**

**6.585.2.1**    **virtual NullPointerException\* de-**  
**caf::lang::exceptions::NullPointerException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p.1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NullPointerException.h**

## 6.586 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2835) is the superclass of classes **Byte** (p. 960), **Double** (p. 1788), **Float** (p. 1904), **Integer** (p. 2077), **Long** (p. 2420), and **Short** (p. 3440).

#include <src/main/decaf/lang/Number.h> Inheritance diagram for decaf::lang::Number:

### Public Member Functions

- virtual `~Number ()`
- virtual unsigned char `byteValue () const`  
*Answers the byte value which the receiver represents.*
- virtual double `doubleValue () const =0`  
*Answers the double value which the receiver represents.*
- virtual float `floatValue () const =0`  
*Answers the float value which the receiver represents.*
- virtual int `intValue () const =0`  
*Answers the int value which the receiver represents.*
- virtual long long `longValue () const =0`  
*Answers the long value which the receiver represents.*
- virtual short `shortValue () const`  
*Answers the short value which the receiver represents.*

### 6.586.1 Detailed Description

The abstract class **Number** (p. 2835) is the superclass of classes **Byte** (p. 960), **Double** (p. 1788), **Float** (p. 1904), **Integer** (p. 2077), **Long** (p. 2420), and **Short** (p. 3440). Subclasses of **Number** (p. 2835) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

### 6.586.2 Constructor & Destructor Documentation

6.586.2.1 virtual decaf::lang::Number::~~Number () [inline, virtual]

### 6.586.3 Member Function Documentation

6.586.3.1 virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

**Returns:**

byte the value of the receiver.

Reimplemented in `decaf::lang::Byte` (p. 962), `decaf::lang::Character` (p. 1102), `decaf::lang::Double` (p. 1790), `decaf::lang::Float` (p. 1906), `decaf::lang::Integer` (p. 2080), `decaf::lang::Long` (p. 2423), and `decaf::lang::Short` (p. 3442).

**6.586.3.2 virtual double decaf::lang::Number::doubleValue () const [pure virtual]**

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implemented in `decaf::lang::Byte` (p. 963), `decaf::lang::Character` (p. 1103), `decaf::lang::Double` (p. 1792), `decaf::lang::Float` (p. 1907), `decaf::lang::Integer` (p. 2081), `decaf::lang::Long` (p. 2424), `decaf::lang::Short` (p. 3443), and `decaf::util::concurrent::atomic::AtomicInteger` (p. 743).

**6.586.3.3 virtual float decaf::lang::Number::floatValue () const [pure virtual]**

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implemented in `decaf::lang::Byte` (p. 964), `decaf::lang::Character` (p. 1103), `decaf::lang::Double` (p. 1793), `decaf::lang::Float` (p. 1909), `decaf::lang::Integer` (p. 2082), `decaf::lang::Long` (p. 2425), `decaf::lang::Short` (p. 3444), and `decaf::util::concurrent::atomic::AtomicInteger` (p. 743).

**6.586.3.4 virtual int decaf::lang::Number::intValue () const [pure virtual]**

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implemented in `decaf::lang::Byte` (p. 964), `decaf::lang::Character` (p. 1104), `decaf::lang::Double` (p. 1793), `decaf::lang::Float` (p. 1909), `decaf::lang::Integer` (p. 2082), `decaf::lang::Long` (p. 2425), `decaf::lang::Short` (p. 3444), and `decaf::util::concurrent::atomic::AtomicInteger` (p. 745).

**6.586.3.5 virtual long long decaf::lang::Number::longValue () const [pure virtual]**

Answers the long value which the receiver represents.

**Returns:**

long long the value of the receiver.

Implemented in `decaf::lang::Byte` (p. 964), `decaf::lang::Character` (p. 1105), `decaf::lang::Double` (p. 1794), `decaf::lang::Float` (p. 1910), `decaf::lang::Integer` (p. 2083), `decaf::lang::Long` (p. 2426), `decaf::lang::Short` (p. 3444), and `decaf::util::concurrent::atomic::AtomicInteger` (p. 745).

#### 6.586.3.6 virtual short decaf::lang::Number::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

##### Returns:

short the value of the receiver.

Reimplemented in `decaf::lang::Byte` (p. 966), `decaf::lang::Character` (p. 1106), `decaf::lang::Double` (p. 1796), `decaf::lang::Float` (p. 1912), `decaf::lang::Integer` (p. 2087), `decaf::lang::Long` (p. 2430), and `decaf::lang::Short` (p. 3446).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

## 6.587 decaf::lang::exceptions::NumberFormatException

### Class Reference

#include <src/main/decaf/lang/exceptions/NumberFormatException.h> Inheritance diagram for decaf::lang::exceptions::NumberFormatException:

#### Public Member Functions

- **NumberFormatException** ()  
*Default Constructor.*
- **NumberFormatException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()  
*Copy Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NumberFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NumberFormatException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NumberFormatException** () throw ()

#### 6.587.1 Constructor & Destructor Documentation

##### 6.587.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException () [inline]

Default Constructor.

Referenced by clone().

##### 6.587.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.587.1.3 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The **Exception** (p. 1831) whose data is to be copied into this one.

**6.587.1.4 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

**6.587.1.5 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.587.1.6 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

**6.587.1.7** **virtual**  
`decaf::lang::exceptions::NumberFormatException::~~NumberFormatException`  
`() throw ()` [inline, virtual]

## 6.587.2 Member Function Documentation

**6.587.2.1** **virtual** `NumberFormatException*` `decaf::lang::exceptions::NumberFormatException::clone ()`  
`const` [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from `decaf::lang::Exception` (p. 1834).

References `NumberFormatException()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NumberFormatException.h`



## 6.588 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

`#include <src/main/cms/ObjectMessage.h>`Inheritance diagram for cms::ObjectMessage:

### Public Member Functions

- virtual `~ObjectMessage ()`

#### 6.588.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized `ObjectMessage` (p. 2841)s.

Since:

1.0

#### 6.588.2 Constructor & Destructor Documentation

##### 6.588.2.1 virtual cms::ObjectMessage::~ObjectMessage () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/ObjectMessage.h`

## 6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Provides an SSLContext that wraps the OpenSSL API.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLContextSpi:

### Public Member Functions

- **OpenSSLContextSpi ()**
- virtual **~OpenSSLContextSpi ()**
- virtual void **providerInit (security::SecureRandom \*random)**  
*Perform the initialization of this Context.*  
**Parameters:**  
*random* Pointer to an instance of a secure random number generator.  
**Exceptions:**  
*NullPointerException* if the SecureRandom instance is NULL.  
*KeyManagementException* if an error occurs while initializing the context.
- virtual **decaf::net::SocketFactory \* providerGetSocketFactory ()**  
*Returns a **SocketFactory** (p. 3524) instance that can be used to create new **SSLSocket** (p. 3563) objects.*  
*The **SocketFactory** (p. 3524) is owned by the Service Provider and should not be destroyed by the caller.*  
**Returns:**  
*SocketFactory* (p. 3524) instance that can be used to create new SSLSockets.  
**Exceptions:**  
*IllegalStateException* if the **SSLContextSpi** (p. 3548) object requires initialization but has not been initialized yet.
- virtual **decaf::net::ServerSocketFactory \* providerGetServerSocketFactory ()**  
*Returns a **ServerSocketFactory** (p. 3361) instance that can be used to create new **SSLServerSocket** (p. 3554) objects.*  
*The **ServerSocketFactory** (p. 3361) is owned by the Service Provider and should not be destroyed by the caller.*  
**Returns:**  
*SocketFactory* (p. 3524) instance that can be used to create new SSLServerSockets.  
**Exceptions:**  
*IllegalStateException* if the **SSLContextSpi** (p. 3548) object requires initialization but has not been initialized yet.

### Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

## 6.589.1 Detailed Description

Provides an SSLContext that wraps the OpenSSL API.

**Since:**

1.0

## 6.589.2 Constructor & Destructor Documentation

**6.589.2.1** decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi()  
( )

**6.589.2.2** virtual decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~~OpenSSLContextSpi()  
( ) [virtual]

## 6.589.3 Member Function Documentation

**6.589.3.1** virtual decaf::net::ServerSocketFactory\* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory()  
( ) [virtual]

Returns a **ServerSocketFactory** (p. 3361) instance that can be used to create new **SSLServerSocket** (p. 3554) objects.

The **ServerSocketFactory** (p. 3361) is owned by the Service Provider and should not be destroyed by the caller.

**Returns:**

**SocketFactory** (p. 3524) instance that can be used to create new SSLServerSockets.

**Exceptions:**

*IllegalStateException* if the **SSLContextSpi** (p. 3548) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p. 3549).

**6.589.3.2** virtual decaf::net::SocketFactory\* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetSocketFactory()  
( ) [virtual]

Returns a **SocketFactory** (p. 3524) instance that can be used to create new **SSLSocket** (p. 3563) objects.

The **SocketFactory** (p. 3524) is owned by the Service Provider and should not be destroyed by the caller.

**Returns:**

**SocketFactory** (p. 3524) instance that can be used to create new SSLSockets.

**Exceptions:**

***IllegalStateException*** if the **SSLContextSpi** (p.3548) object requires initialization but has not been initialized yet.

Implements **decaf::net::ssl::SSLContextSpi** (p.3549).

**6.589.3.3** virtual void decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerInit (security::SecureRandom \* *random*) [virtual]

Perform the initialization of this Context.

**Parameters:**

*random* Pointer to an instance of a secure random number generator.

**Exceptions:**

***NullPointerException*** if the SecureRandom instance is NULL.

***KeyManagementException*** if an error occurs while initializing the context.

Implements **decaf::net::ssl::SSLContextSpi** (p.3550).

**6.589.4 Friends And Related Function Documentation**

**6.589.4.1** friend class OpenSSLSocket [friend]

**6.589.4.2** friend class OpenSSLSocketFactory [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLContextSpi.h**

## 6.590 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference

Container class for parameters that are Common to OpenSSL socket classes.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>
```

### Public Member Functions

- virtual `~OpenSSLParameters ()`
- bool `getNeedClientAuth ()` const
- void `setNeedClientAuth (bool value)`
- bool `getWantClientAuth ()` const
- void `setWantClientAuth (bool value)`
- bool `getUseClientMode ()` const
- void `setUseClientMode (bool value)`
- std::vector< std::string > `getSupportedCipherSuites ()` const
- std::vector< std::string > `getSupportedProtocols ()` const
- std::vector< std::string > `getEnabledCipherSuites ()` const
- void `setEnabledCipherSuites (const std::vector< std::string > &suites)`
- std::vector< std::string > `getEnabledProtocols ()` const
- void `setEnabledProtocols (const std::vector< std::string > &protocols)`
- `OpenSSLParameters * clone ()` const

*Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.*

### 6.590.1 Detailed Description

Container class for parameters that are Common to OpenSSL socket classes.

Since:

1.0

### 6.590.2 Constructor & Destructor Documentation

- 6.590.2.1** virtual  
**decaf::internal::net::ssl::openssl::OpenSSLParameters::~~OpenSSLParameters**  
 () [virtual]

### 6.590.3 Member Function Documentation

- 6.590.3.1** `OpenSSLParameters* de-`  
**caf::internal::net::ssl::openssl::OpenSSLParameters::clone**  
 () const

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.

- 6.590.3.2 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites () const`
- 6.590.3.3 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols () const`
- 6.590.3.4 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth () const [inline]`
- 6.590.3.5 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites () const`
- 6.590.3.6 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols () const`
- 6.590.3.7 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode () const [inline]`
- 6.590.3.8 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth () const [inline]`
- 6.590.3.9 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites (const std::vector< std::string > & suites)`
- 6.590.3.10 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols (const std::vector< std::string > & protocols)`
- 6.590.3.11 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth (bool value) [inline]`
- 6.590.3.12 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode (bool value) [inline]`
- 6.590.3.13 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth (bool value) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h`

## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

SSLServerSocket based on OpenSSL library [code](#) (p. 1183).

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocket:

### Public Member Functions

- **OpenSSLServerSocket** (OpenSSLParameters \*parameters)
- virtual ~**OpenSSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3554).  
Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).*  
**Returns:**  
*a vector containing the names of all the supported cipher suites.*
- virtual std::vector< std::string > **getSupportedProtocols** () const  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3554) instance.*  
**Returns:**  
*a vector containing the names of all the supported protocols.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3554).*  
**Returns:**  
*vector of the names of all enabled Cipher Suites.*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)  
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3554) connection.  
Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*  
**Parameters:**  
*suites An Vector of names for all the Cipher Suites that are to be enabled.*  
**Exceptions:**  
*IllegalArgumentException if the vector is empty or one of the names is invalid.*
- virtual std::vector< std::string > **getEnabledProtocols** () const  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3554).*  
**Returns:**  
*vector of the names of all enabled Protocols.*

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)  
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3554) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*  
**Parameters:**  
*protocols* An Vector of names for all the Protocols that are to be enabled.  
**Exceptions:**  
*IllegalArgumentException* if the vector is empty or one of the names is invalid.
- virtual bool **getWantClientAuth** () const  
**Returns:**  
*true if the **Socket** (p. 3503) request client Authentication.*
- virtual void **setWantClientAuth** (bool value)  
*Sets whether or not this **Socket** (p. 3503) will request Client Authentication. If set to true the **Socket** (p. 3503) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.*  
**Parameters:**  
*value* Whether the server socket should request client authentication.
- virtual bool **getNeedClientAuth** () const  
**Returns:**  
*true if the **Socket** (p. 3503) requires client Authentication.*
- virtual void **setNeedClientAuth** (bool value)  
*Sets whether or not this **Socket** (p. 3503) will require Client Authentication. If set to true the **Socket** (p. 3503) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.*  
**Parameters:**  
*value* Whether the server socket should require client authentication.
- virtual **decaf::net::Socket \* accept** () throw ( decaf::io::IOException )  
*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3352), the caller blocks until a connection is made. If the `SO_TIMEOUT` option is set this method could throw a **SocketTimeoutException** (p. 3542) if the operation times out.*  
**Returns:**  
*a new **Socket** (p. 3503) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.*  
**Exceptions:**  
*IOException* if an I/O error occurs while binding the socket.  
*SocketException* (p. 3521) if an error occurs while blocking on the accept call.  
*SocketTimeoutException* (p. 3542) if the `SO_TIMEOUT` option was used and the accept timed out.

## 6.591.1 Detailed Description

SSLServerSocket based on OpenSSL library **code** (p. 1183).



Since:

1.0

## 6.591.2 Constructor & Destructor Documentation

**6.591.2.1** decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket (OpenSSLParameters \* *parameters*)

**6.591.2.2** virtual  
decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~~OpenSSLServerSocket  
( ) [virtual]

## 6.591.3 Member Function Documentation

**6.591.3.1** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept ( ) throw ( decaf::io::IOException ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3352), the caller blocks until a connection is made.

If the SO\_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3542) if the operation times out.

**Returns:**

a new **Socket** (p. 3503) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

**Exceptions:**

**IOException** if an I/O error occurs while binding the socket.

**SocketException** (p. 3521) if an error occurs while blocking on the accept call.

**SocketTimeoutException** (p. 3542) if the SO\_TIMEOUT option was used and the accept timed out.

Reimplemented from **decaf::net::ServerSocket** (p. 3356).

**6.591.3.2** virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites ( ) const [virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3554).

**Returns:**

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3556).

**6.591.3.3** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols() const [virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3554).

**Returns:**

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3557).

**6.591.3.4** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth() const [virtual]`

**Returns:**

true if the **Socket** (p. 3503) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3557).

**6.591.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites() const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3554).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).

**Returns:**

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3557).

**6.591.3.6** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols() const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3554) instance.

**Returns:**

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3557).

**6.591.3.7** virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth() const [virtual]

**Returns:**

true if the **Socket** (p. 3503) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3558).

**6.591.3.8** virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites(const std::vector< std::string > & suites) [virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3554) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters:**

*suites* An Vector of names for all the Cipher Suites that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3558).

**6.591.3.9** virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols(const std::vector< std::string > & protocols) [virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3554) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters:**

*protocols* An Vector of names for all the Protocols that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3558).

**6.591.3.10** virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth(bool value) [virtual]

Sets whether or not this **Socket** (p. 3503) will require Client Authentication.

If set to true the **Socket** (p. 3503) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

**Parameters:**

*value* Whether the server socket should require client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3558).

**6.591.3.11**    **virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth (bool *value*)** [virtual]

Sets whether or not this **Socket** (p. 3503) will request Client Authentication.

If set to true the **Socket** (p. 3503) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

**Parameters:**

*value* Whether the server socket should request client authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3559).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h`

## 6.592 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory:

### Public Member Functions

- **OpenSSLServerSocketFactory (OpenSSLContextSpi \*parent)**
- **virtual ~OpenSSLServerSocketFactory ()**
- **virtual decaf::net::ServerSocket \* createServerSocket ()**

Create a new **ServerSocket** (p. 3352) that is unbound.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.

- **virtual decaf::net::ServerSocket \* createServerSocket (int port)**

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.

- **virtual decaf::net::ServerSocket \* createServerSocket (int port, int backlog)**

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.

**backlog** The number of pending connect request the **ServerSocket** (p. 3352) can queue.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.

- **virtual decaf::net::ServerSocket \* createServerSocket (int port, int backlog, const decaf::net::InetAddress \*address)**

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is *NULL* than the **ServerSocket** (p. 3352) will listen on all interfaces.

**Parameters:**

**port** The port to bind the **ServerSocket** (p. 3352) to.  
**backlog** The number of pending connect request the **ServerSocket** (p. 3352) can queue.  
**address** The address of the interface on the local machine to bind to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

**IOException** if the **ServerSocket** (p. 3352) cannot be created for some reason.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

**getSupportedCipherSuites()** (p. 3561)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

**getDefaultCipherSuites()** (p. 3561)

## 6.592.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

**Since:**

1.0

## 6.592.2 Constructor & Destructor Documentation

**6.592.2.1** `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::OpenSSLServerSocketFactory(OpenSSLContextSpi * parent)`

**6.592.2.2** `virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::~~OpenSSLServerSocketFactory() [virtual]`

## 6.592.3 Member Function Documentation

**6.592.3.1** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket(int port, int backlog, const decaf::net::InetAddress * address) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3352) will listen on all interfaces.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

*address* The address of the interface on the local machine to bind to.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.592.3.2** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket(int port, int backlog) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3362).

**6.592.3.3** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket(int port) [virtual]`

Create a new **ServerSocket** (p. 3352) that is bound to the given port.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Parameters:**

*port* The port to bind the **ServerSocket** (p. 3352) to.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implements **decaf::net::ServerSocketFactory** (p. 3363).

**6.592.3.4** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket() [virtual]`

Create a new **ServerSocket** (p. 3352) that is unbound.

The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

**Returns:**

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

**Exceptions:**

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3363).

**6.592.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites() [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).



**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

`getSupportedCipherSuites()` (p. 3561)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3561).

**6.592.3.6 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites() [virtual]**

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

`getDefaultCipherSuites()` (p. 3561)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3561).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h`

## 6.593 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

#include <src/main/decaf/internal/net/ssl/openssl/0penSSLSocket.h>Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

### Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** \*parameters)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- virtual ~**OpenSSLSocket** ()
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

*Connects to the specified destination, with a specified timeout value.*

*If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3542) is thrown. A timeout value of zero is treated as an infinite timeout.*

**Parameters:**

*host* The host name or IP address of the remote host to connect to.

*port* The port on the remote host to connect to.

*timeout* The number of Milliseconds to wait before treating the connection as failed.

**Exceptions:**

**IOException** Thrown if a failure occurred in the connect.

**SocketTimeoutException** (p. 3542) if the timeout for connection is exceeded.

**IllegalArgumentException** if the timeout value is negative or the endpoint is invalid.

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **Socket** (p. 3503).*

*Once closed a **Socket** (p. 3503) cannot be connected or otherwise operated upon, a new **Socket** (p. 3503) instance must be created.*

**Exceptions:**

**IOException** if an I/O error occurs while closing the **Socket** (p. 3503).

- virtual **decaf::io::InputStream** \* **getInputStream** () throw ( decaf::io::IOException )

*Gets the **InputStream** for this socket if its connected.*

*The pointer returned is the property of the associated **Socket** (p. 3503) and should not be deleted by the caller.*

*When the returned **InputStream** is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure. Closing the **InputStream** will also close the underlying **Socket** (p. 3503).*

**Returns:**

*The InputStream for this socket.*

**Exceptions:**

**IOException** if an error occurs during creation of the *InputStream*, also if the **Socket** (p. 3503) is not connected or the input has been shutdown previously.

- virtual **decaf::io::OutputStream \* getOutputStream ()** throw ( decaf::io::IOException )

*Gets the OutputStream for this socket if it is connected.*

*The pointer returned is the property of the **Socket** (p. 3503) instance and should not be deleted by the caller.*

*Closing the returned **Socket** (p. 3503) will also close the underlying **Socket** (p. 3503).*

**Returns:**

*the OutputStream for this socket.*

**Exceptions:**

**IOException** if an error occurs during the creation of this *OutputStream*, or if the **Socket** (p. 3503) is closed or the output has been shutdown previously.

- virtual void **shutdownInput ()** throw ( decaf::io::IOException )

*Shuts down the InputStream for this socket essentially marking it as EOF.*

*The stream returns EOF for any calls to read after this method has been called.*

**Exceptions:**

**IOException** if an I/O error occurs while performing this operation.

- virtual void **shutdownOutput ()** throw ( decaf::io::IOException )

*Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to *OutputStream::write* will throw an *IOException*.*

**Exceptions:**

**IOException** if an I/O error occurs while performing this operation.

- virtual void **setOOBInline** (bool value) throw ( decaf::net::SocketException )

*Sets the value of the *OOBINLINE* for this socket, by default this option is disabled.*

*If enabled the urgent data is read inline on the *Socket*'s *InputStream*, no notification is give.*

**Returns:**

*true if *OOBINLINE* is enabled, false otherwise.*

**Exceptions:**

**SocketException** (p. 3521) if an error is encountered while performing this operation.

- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )

*Sends on byte of urgent data to the **Socket** (p. 3503).*

**Parameters:**

**data** *The value to write as urgent data, only the lower eight bits are sent.*

**Exceptions:**

**IOException** if an I/O error occurs while performing this operation.

- virtual std::vector< std::string > **getSupportedCipherSuites ()** const

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3563).*

*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).*

**Returns:**

*a vector containing the names of all the supported cipher suites.*

- virtual std::vector< std::string > **getSupportedProtocols** () const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3563) instance.*

**Returns:**

*a vector containing the names of all the supported protocols.*

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3503).*

**Returns:**

*vector of the names of all enabled Cipher Suites.*

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3503) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

**Parameters:**

***suites** An Vector of names for all the Cipher Suites that are to be enabled.*

**Exceptions:**

***IllegalArgumentException** if the vector is empty or one of the names is invalid.*

- virtual std::vector< std::string > **getEnabledProtocols** () const

*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3503).*

**Returns:**

*vector of the names of all enabled Protocols.*

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3503) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*

**Parameters:**

***protocols** An Vector of names for all the Protocols that are to be enabled.*

**Exceptions:**

***IllegalArgumentException** if the vector is empty or one of the names is invalid.*

- virtual void **startHandshake** ()

*Initiates a handshake for this **SSL Connection**, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*

*When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an **IOException** to indicate an error.*

**Exceptions:**

***IOException** if an I/O error occurs while performing the Handshake*

- virtual void **setUseClientMode** (bool value)

*Determines the mode that the socket uses when a handshake is initiated, client or server.*

*This method must be called prior to any handshake attempts on this **Socket** (p. 3503), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.*

**Parameters:**

*value* The mode setting, true for client or false for server.

**Exceptions:**

***IllegalArgumentException** if the handshake process has begun and mode is locked.*

- virtual bool **getUseClientMode** () const

*Gets whether this **Socket** (p. 3503) is in Client or Server mode, true indicates that the mode is set to Client.*

**Returns:**

*true if the **Socket** (p. 3503) is in Client mode, false otherwise.*

- virtual void **setNeedClientAuth** (bool value)

*Sets the **Socket** (p. 3503) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

*This option only applies to sockets in the Server mode.*

*If the option is enabled and the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the **setWantClientAuth** method.*

**Parameters:**

*value* The value indicating if a client is required to authenticate itself or not.

- virtual bool **getNeedClientAuth** () const

*Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.*

*This option is only useful when the socket is operating in server mode.*

**Returns:**

*true if client authentication is required.*

- virtual void **setWantClientAuth** (bool value)

*Sets the **Socket** (p. 3503) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

*This option only applies to sockets in the Server mode.*

*If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid then the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the **setNeedClientAuth** method.*

**Parameters:**

*value* The value indicating if a client is requested to authenticate itself or not.

- virtual bool **getWantClientAuth** () const

*Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.*

*This option is only useful when the socket is operating in server mode.*

**Returns:**

*true if client authentication is required.*

- `int read` (unsigned char \*buffer, int size, int offset, int length)  
*Reads the requested data from the Socket and write it into the passed in buffer.*
- `void write` (const unsigned char \*buffer, int size, int offset, int length)  
*Writes the specified data in the passed in buffer to the Socket.*
- `int available` ()  
*Gets the number of bytes in the Socket buffer that can be read without blocking.*

### 6.593.1 Detailed Description

Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

Since:

1.0

### 6.593.2 Constructor & Destructor Documentation

- 6.593.2.1** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket`  
(OpenSSLParameters \* *parameters*)
- 6.593.2.2** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket`  
(OpenSSLParameters \* *parameters*, const decaf::net::InetAddress \* *address*, int *port*)
- 6.593.2.3** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket`  
(OpenSSLParameters \* *parameters*, const decaf::net::InetAddress \* *address*, int *port*, const decaf::net::InetAddress \* *localAddress*, int *localPort*)
- 6.593.2.4** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket`  
(OpenSSLParameters \* *parameters*, const std::string & *host*, int *port*)
- 6.593.2.5** `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket`  
(OpenSSLParameters \* *parameters*, const std::string & *host*, int *port*, const decaf::net::InetAddress \* *localAddress*, int *localPort*)
- 6.593.2.6** `virtual`  
`decaf::internal::net::ssl::openssl::OpenSSLSocket::~~OpenSSLSocket` ()  
[virtual]

### 6.593.3 Member Function Documentation

- 6.593.3.1** `int decaf::internal::net::ssl::openssl::OpenSSLSocket::available` ()

Gets the number of bytes in the Socket buffer that can be read without blocking.

**Returns:**

the number of bytes that can be read from the Socket without blocking.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

**6.593.3.2** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close ()  
throw ( decaf::io::IOException ) [virtual]`

Closes the **Socket** (p. 3503).

Once closed a **Socket** (p. 3503) cannot be connected or otherwise operated upon, a new **Socket** (p. 3503) instance must be created.

**Exceptions:**

*IOException* if an I/O error occurs while closing the **Socket** (p. 3503).

Reimplemented from **decaf::net::Socket** (p. 3509).

**6.593.3.3** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect  
(const std::string & host, int port, int timeout) throw (  
decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException  
) [virtual]`

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3542) is thrown. A timeout value of zero is treated as an infinite timeout.

**Parameters:**

*host* The host name or IP address of the remote host to connect to.

*port* The port on the remote host to connect to.

*timeout* The number of Milliseconds to wait before treating the connection as failed.

**Exceptions:**

*IOException* Thrown if a failure occurred in the connect.

*SocketTimeoutException* (p. 3542) if the timeout for connection is exceeded.

*IllegalArgumentException* if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 3509).

**6.593.3.4** `virtual std::vector<std::string> de-  
caf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites ()  
const [virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3503).

**Returns:**

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3566).

**6.593.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols () const [virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 3503).

**Returns:**

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3566).

**6.593.3.6** `virtual decaf::io::InputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getInputStream () throw (decaf::io::IOException) [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3503) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3503).

**Returns:**

The InputStream for this socket.

**Exceptions:**

**IOException** if an error occurs during creation of the InputStream, also if the **Socket** (p. 3503) is not connected or the input has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 3510).

**6.593.3.7** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeedClientAuth () const [virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

**Returns:**

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3567).



**6.593.3.8** `virtual decaf::io::OutputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getOutputStream () throw ( decaf::io::IOException ) [virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3503) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3503) will also close the underlying **Socket** (p. 3503).

**Returns:**

the OutputStream for this socket.

**Exceptions:**

*IOException* if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 3503) is closed or the output has been shutdown previously.

Reimplemented from **decaf::net::Socket** (p. 3511).

**6.593.3.9** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites () const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3563).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).

**Returns:**

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3567).

**6.593.3.10** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols () const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3563) instance.

**Returns:**

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3567).

**6.593.3.11** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode () const [virtual]`

Gets whether this **Socket** (p. 3503) is in Client or Server mode, true indicates that the mode is set to Client.

**Returns:**

true if the **Socket** (p. 3503) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 3568).

**6.593.3.12** **virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getWantClientAuth ()**  
**const** [virtual]

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

**Returns:**

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3568).

**6.593.3.13** **int decaf::internal::net::ssl::openssl::OpenSSLSocket::read (unsigned char \* *buffer*, int *size*, int *offset*, int *length*)**

Reads the requested data from the Socket and write it into the passed in buffer.

**Parameters:**

*buffer* The buffer to read into  
*size* The size of the specified buffer  
*offset* The offset into the buffer where reading should start filling.  
*length* The number of bytes past offset to fill with data.

**Returns:**

the actual number of bytes read or -1 if at EOF.

**Exceptions:**

*IOException* if an I/O error occurs during the read.  
*NullPointerException* if buffer is Null.  
*IndexOutOfBoundsException* if offset + length is greater than buffer size.

**6.593.3.14** **virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::sendUrgentData (int *data*)** throw ( decaf::io::IOException ) [virtual]

Sends one byte of urgent data to the **Socket** (p. 3503).

**Parameters:**

*data* The value to write as urgent data, only the lower eight bits are sent.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3514).

**6.593.3.15** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites (const std::vector< std::string > & suites) [virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3503) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters:**

*suites* An Vector of names for all the Cipher Suites that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 3568).

**6.593.3.16** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3503) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters:**

*protocols* An Vector of names for all the Protocols that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implements **decaf::net::ssl::SSLSocket** (p. 3568).

**6.593.3.17** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setNeedClientAuth (bool value) [virtual]`

Sets the **Socket** (p. 3503) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the `setWantClientAuth` method.

**Parameters:**

*value* The value indicating if a client is required to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 3569).

**6.593.3.18** **virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOBInline (bool *value*) throw ( decaf::net::SocketException )** [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

**Returns:**

true if OOBINLINE is enabled, false otherwise.

**Exceptions:**

*SocketException* (p. 3521) if an error is encountered while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3515).

**6.593.3.19** **virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setUseClientMode (bool *value*)** [virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3503), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

**Parameters:**

*value* The mode setting, true for client or false for server.

**Exceptions:**

*IllegalArgumentException* if the handshake process has begun and mode is locked.

Implements **decaf::net::ssl::SSLSocket** (p. 3569).

**6.593.3.20** **virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth (bool *value*)** [virtual]

Sets the **Socket** (p. 3503) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

**Parameters:**

*value* The value indicating if a client is requested to authenticate itself or not.

Implements **decaf::net::ssl::SSLSocket** (p. 3570).

**6.593.3.21** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput () throw ( decaf::io::IOException ) [virtual]`

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3517).

**6.593.3.22** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput () throw ( decaf::io::IOException ) [virtual]`

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Reimplemented from **decaf::net::Socket** (p. 3518).

**6.593.3.23** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::startHandshake () [virtual]`

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an IOException to indicate an error.

**Exceptions:**

*IOException* if an I/O error occurs while performing the Handshake

Implements **decaf::net::ssl::SSLSocket** (p. 3570).

**6.593.3.24** `void decaf::internal::net::ssl::openssl::OpenSSLSocket::write (const unsigned char * buffer, int size, int offset, int length)`

Writes the specified data in the passed in buffer to the Socket.

**Parameters:**

- buffer* The buffer to write to the socket.
- size* The size of the specified buffer.
- offset* The offset into the buffer where the data to write starts at.
- length* The number of bytes past offset to write.

**Exceptions:**

- IOException*** if an I/O error occurs during the write.
- NullPointerException*** if buffer is Null.
- IndexOutOfBoundsException*** if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h`

## 6.594 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketException:

### Public Member Functions

- **OpenSSLSocketException** () throw ()  
*Creates a new **OpenSSLSocketException** (p. 2871) with default values.*
- **OpenSSLSocketException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **OpenSSLSocketException** (const **OpenSSLSocketException** &ex) throw ()  
*Copy Constructor.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Create a new **OpenSSLSocketException** (p. 2871) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const std::exception \*cause) throw ()  
*Creates a new **OpenSSLSocketException** (p. 2871) with the passed exception set as the cause of this exception.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Create a new **OpenSSLSocketException** (p. 2871) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber) throw ()  
*Create a new **OpenSSLSocketException** (p. 2871) and initializes the file name and line number where this message occurred.*
- virtual **OpenSSLSocketException** \* clone () const  
*Clones this exception.*
- virtual ~**OpenSSLSocketException** () throw ()

### Protected Member Functions

- std::string **getErrorString** () const  
*Gets and formats an error message string from the OpenSSL error stack.*

### 6.594.1 Detailed Description

Subclass of the standard `SocketException` that knows how to produce an error message from the OpenSSL error stack.

**Since:**

1.0

### 6.594.2 Constructor & Destructor Documentation

#### 6.594.2.1 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException() throw ()`

Creates an new **OpenSSLSocketException** (p.2871) with default values.

#### 6.594.2.2 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException(const Exception & ex) throw ()`

Conversion Constructor from some other Exception.

**Parameters:**

*ex* An Exception object that should become this type of Exception.

#### 6.594.2.3 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException(const OpenSSLSocketException & ex) throw ()`

Copy Constructor.

**Parameters:**

*ex* The **OpenSSLSocketException** (p.2871) whose values should be copied to this instance.

#### 6.594.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()`

Create a new **OpenSSLSocketException** (p.2871) and initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message.

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown (can be null).

*msg* The error message to report.

*...* The list of primitives that are formatted into the message.



**6.594.2.5 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const std::exception \* *cause*) throw ()**

Creates a new **OpenSSLSocketException** (p. 2871) with the passed exception set as the cause of this exception.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.594.2.6 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw ()**

Create a new **OpenSSLSocketException** (p. 2871) and initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message.

**Parameters:**

*file* The file name where exception occurs.  
*lineNumber* The line number where the exception occurred.  
*msg* The error message to report.  
... The list of primitives that are formatted into the message

**6.594.2.7 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException (const char \* *file*, const int *lineNumber*) throw ()**

Create a new **OpenSSLSocketException** (p. 2871) and initializes the file name and line number where this message occurred. Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

**Parameters:**

*file* The file name where exception occurs.  
*lineNumber* The line number where the exception occurred.

**6.594.2.8 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketException::~~OpenSSLSocketException () throw () [virtual]****6.594.3 Member Function Documentation****6.594.3.1 virtual OpenSSLSocketException\* decaf::internal::net::ssl::openssl::OpenSSLSocketException::clone () const [inline, virtual]**

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override this method.

**Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3522).

**6.594.3.2** `std::string decaf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString()` **const** [protected]

Gets and formats an error message string from the OpenSSL error stack.

**Returns:**

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h`

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketFactory:

### Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** \*parent)
- virtual **~OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3503) object.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if the **Socket** (p. 3503) cannot be created.*
- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
**Parameters:**  
*host The host to connect the socket to.*  
*port The port on the remote host to connect to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.*  
***UnknownHostException** (p. 3907) if the host name is not known.*
- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*  
*The **Socket** (p. 3503) will be bound to the specified local address and port.*  
**Parameters:**  
*host The host to connect the socket to.*  
*port The port on the remote host to connect to.*  
*ifAddress The address on the local machine to bind the **Socket** (p. 3503) to.*  
*localPort The local port to bind the **Socket** (p. 3503) to.*  
**Returns:**  
*a new **Socket** (p. 3503) object, caller must free this object when done.*  
**Exceptions:**  
***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.*  
***UnknownHostException** (p. 3907) if the host name is not known.*

- virtual **decaf::net::Socket \* createSocket** (const std::string &hostname, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*

**Parameters:**

***host** The host name or IP address to connect the socket to.  
**port** The port on the remote host to connect to.*

**Returns:**

*a new **Socket** (p. 3503) object, caller must free this object when done.*

**Exceptions:**

***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.*

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*

**Parameters:**

***host** The host name or IP address to connect the socket to.  
**port** The port on the remote host to connect to.  
**ifAddress** The address on the local machine to bind the **Socket** (p. 3503) to.  
**localPort** The local port to bind the **Socket** (p. 3503) to.*

**Returns:**

*a new **Socket** (p. 3503) object, caller must free this object when done.*

**Exceptions:**

***IOException** if an I/O error occurs while creating the **Socket** (p. 3503) object.  
**UnknownHostException** (p. 3907) if the host name is not known.*

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

*Returns the list of cipher suites which are enabled by default.*

*Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).*

**Returns:**

*an STL vector containing the list of cipher suites enabled by default.*

**See also:**

***getSupportedCipherSuites()** (p. 3573)*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*

*Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.*

**Returns:**

*an STL vector containing the list of supported cipher suites.*

**See also:**

***getDefaultCipherSuites()** (p. 3573)*

- virtual **decaf::net::Socket \* createSocket** (**decaf::net::Socket** \*socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port. This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters:**

*socket* The existing socket to layer over.  
*host* The server host the original **Socket** (p. 3503) is connected to.  
*port* The server port the original **Socket** (p. 3503) is connected to.  
*autoClose* Should the layered over **Socket** (p. 3503) be closed when the topmost socket is closed.

**Returns:**

a new **Socket** (p. 3503) instance that wraps the given **Socket** (p. 3503).

**Exceptions:**

*IOException* if an I/O exception occurs while performing this operation.  
*UnknownHostException* (p. 3907) if the host is unknown.

## 6.595.1 Detailed Description

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

**Since:**

1.0

## 6.595.2 Constructor & Destructor Documentation

**6.595.2.1** decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory (OpenSSLContextSpi \* *parent*)

**6.595.2.2** virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~~OpenSSLSocketFactory () [virtual]

## 6.595.3 Member Function Documentation

**6.595.3.1** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (decaf::net::Socket \* *socket*, std::string *host*, int *port*, bool *autoClose*) [virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters:**

*socket* The existing socket to layer over.  
*host* The server host the original **Socket** (p. 3503) is connected to.  
*port* The server port the original **Socket** (p. 3503) is connected to.  
*autoClose* Should the layered over **Socket** (p. 3503) be closed when the topmost socket is closed.

**Returns:**

a new **Socket** (p. 3503) instance that wraps the given **Socket** (p. 3503).

**Exceptions:**

***IOException*** if an I/O exception occurs while performing this operation.

***UnknownHostException*** (p. 3907) if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3572).

```
6.595.3.2 virtual decaf::net::Socket* de-
caf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
(const std::string & name, int port, const decaf::net::InetAddress
* ifAddress, int localPort) throw ( decaf::io::IOException,
decaf::net::UnknownHostException ) [virtual]
```

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

```
6.595.3.3 virtual decaf::net::Socket* de-
caf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
(const std::string & hostname, int port) throw ( decaf::io::IOException,
decaf::net::UnknownHostException ) [virtual]
```

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3525).

**6.595.3.4** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*, const decaf::net::InetAddress \* *ifAddress*, int *localPort*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

The **Socket** (p. 3503) will be bound to the specified local address and port.

**Parameters:**

***host*** The host to connect the socket to.

***port*** The port on the remote host to connect to.

***ifAddress*** The address on the local machine to bind the **Socket** (p. 3503) to.

***localPort*** The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.595.3.5** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket (const decaf::net::InetAddress \* *host*, int *port*) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

***host*** The host to connect the socket to.

***port*** The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if an I/O error occurs while creating the **Socket** (p. 3503) object.

***UnknownHostException*** (p. 3907) if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3526).

**6.595.3.6** `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket () throw ( decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3503) object.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

***IOException*** if the **Socket** (p. 3503) cannot be created.

Reimplemented from **decaf::net::SocketFactory** (p. 3527).

**6.595.3.7** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites () [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns:**

an STL vector containing the list of cipher suites enabled by default.

**See also:**

**getSupportedCipherSuites()** (p. 3573)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3573).

**6.595.3.8** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getSupportedCipherSuites () [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.



**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

**getDefaultCipherSuites()** (p. 3573)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3573).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h`

## 6.596 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference

An output stream for reading data from an OpenSSL Socket instance.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h> Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream:

### Public Member Functions

- **OpenSSLSocketInputStream** (OpenSSLSocket \*socket)
- virtual ~**OpenSSLSocketInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

**Returns:**

*the number of bytes available on this input stream.*

**Exceptions:**

*IOException (p. 2142) if an I/O error occurs.*

- virtual void **close** () throw ( decaf::io::IOException )
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported.*

### Protected Member Functions

- virtual int **doReadByte** () throw ( io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsExpection, decaf::lang::exceptions::NullPointerExpection )

#### 6.596.1 Detailed Description

An output stream for reading data from an OpenSSL Socket instance.

**Since:**

1.0

## 6.596.2 Constructor & Destructor Documentation

**6.596.2.1** decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::OpenSSLSocketInputStream (OpenSSLSocket \* *socket*)

**6.596.2.2** virtual  
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::~~OpenSSLSocketInputStream () [virtual]

## 6.596.3 Member Function Documentation

**6.596.3.1** virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::available ()  
const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

### Returns:

the number of bytes available on this input stream.

### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.596.3.2** virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close () throw ( decaf::io::IOException ) [virtual]

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 2043) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.596.3.3** virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded (unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2046).

**6.596.3.4** virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte  
( ) throw ( io::IOException ) [protected, virtual]

Implements decaf::io::InputStream (p. 2046).

**6.596.3.5** virtual long long decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip  
(long long *num*) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Not supported. Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

*num* The number of bytes to skip.

#### Returns:

total bytes skipped

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

*UnsupportedOperationException* if the concrete stream class does not support skipping bytes.

Reimplemented from decaf::io::InputStream (p. 2050).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h

## 6.597 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2858) instance.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h> Inheritance  
diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream:

### Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** \*socket)
- virtual ~**OpenSSLSocketOutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

**Exceptions:**

***IOException** (p. 2142) if an error occurs while closing.*

*The default implementation of this method does nothing.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.597.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2858) instance.

**Since:**

1.0

## 6.597.2 Constructor & Destructor Documentation

**6.597.2.1** `decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream (OpenSSLSocket * socket)`

**6.597.2.2** `virtual decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::~~OpenSSLSocketOutputStream ()` [virtual]

## 6.597.3 Member Function Documentation

**6.597.3.1** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ()`  
`throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from `decaf::io::OutputStream` (p. 2909).

**6.597.3.2** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded (const unsigned char * buffer, int size, int offset, int length)` throw ( `decaf::io::IOException`, `decaf::lang::exceptions::NullPointerException`, `decaf::lang::exceptions::IndexOutOfBoundsException` ) [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2909).

**6.597.3.3** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte (unsigned char c)` throw ( `decaf::io::IOException` ) [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2910).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h`

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

### Public Member Functions

- **OpenWireFormat** (const **decaf::util::Properties** &properties)  
*Constructs a new **OpenWireFormat** (p. 2887) object.*
- virtual **~OpenWireFormat** ()
- virtual bool **hasNegotiator** () const  
*Returns true if this **WireFormat** (p. 3976) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual **Pointer< transport::Transport > createNegotiator** (const **Pointer< transport::Transport >** &transport) throw ( **decaf::lang::exceptions::UnsupportedOperationException** )  
*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*
- void **addMarshaller** (**marshal::DataStreamMarshaller** \*marshaller)  
*Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.*
- virtual void **marshal** (const **Pointer< commands::Command >** &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer< commands::Command > unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )  
*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.*
- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** \*object, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Utility method for Tight Marshaling the given object to the boolean stream passed.*
- void **tightMarshalNestedObject2** (**commands::DataStructure** \*o, **decaf::io::DataOutputStream** \*ds, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Utility method that will Tight **marshal** (p. 107) some internally nested object that implements the **DataStructure** interface.*

- **commands::DataStructure** \* **tightUnmarshalNestedObject** (decaf::io::DataInputStream \*dis, utils::BooleanStream \*bs) throw ( decaf::io::IOException )  
*Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.*
- **commands::DataStructure** \* **looseUnmarshalNestedObject** (decaf::io::DataInputStream \*dis) throw ( decaf::io::IOException )  
*Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.*
- void **looseMarshalNestedObject** (commands::DataStructure \*o, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )  
*Utility method to loosely Marshal an object that is derived from the DataStrcutre interface.*
- void **renegotiateWireFormat** (const commands::WireFormatInfo &info) throw ( decaf::lang::exceptions::IllegalStateException )  
*Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.*
- virtual void **setPreferedWireFormatInfo** (const Pointer< commands::WireFormatInfo > &info) throw ( decaf::lang::exceptions::IllegalStateException )  
*Configures this object using the provided WireformatInfo object.*
- virtual const Pointer< commands::WireFormatInfo > & **getPreferedWireFormatInfo** () const  
*Gets the Preferred WireFormatInfo object that this class holds.*
- bool **isStackTraceEnabled** () const  
*Checks if the stackTraceEnabled flag is on.*
- void **setStackTraceEnabled** (bool stackTraceEnabled)  
*Sets if the stackTraceEnabled flag is on.*
- bool **isTcpNoDelayEnabled** () const  
*Checks if the tcpNoDelayEnabled flag is on.*
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)  
*Sets if the tcpNoDelayEnabled flag is on.*
- int **getVersion** () const  
*Get the current Wireformat Version.*
- void **setVersion** (int version) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Set the current Wireformat Version.*
- virtual bool **inReceive** () const  
*Is there a Message being unmarshaled?*
- bool **isCacheEnabled** () const



*Checks if the cacheEnabled flag is on.*

- void **setCacheEnabled** (bool cacheEnabled)  
*Sets if the cacheEnabled flag is on.*
- int **getCacheSize** () const  
*Returns the currently set Cache size.*
- void **setCacheSize** (int value)  
*Sets the current Cache size.*
- bool **isTightEncodingEnabled** () const  
*Checks if the tightEncodingEnabled flag is on.*
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)  
*Sets if the tightEncodingEnabled flag is on.*
- bool **isSizePrefixDisabled** () const  
*Checks if the sizePrefixDisabled flag is on.*
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)  
*Sets if the sizePrefixDisabled flag is on.*
- long long **getMaxInactivityDuration** () const  
*Gets the MaxInactivityDuration setting.*
- void **setMaxInactivityDuration** (long long value)  
*Sets the MaxInactivityDuration setting.*
- long long **getMaxInactivityDurationInitialDelay** () const  
*Gets the MaxInactivityDurationInitialDelay setting.*
- void **setMaxInactivityDurationInitialDelay** (long long value)  
*Sets the MaxInactivityDurationInitialDelay setting.*

## Protected Member Functions

- **commands::DataStructure \* doUnmarshal** (decaf::io::DataInputStream \*dis)  
throw ( decaf::io::IOException )  
*Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.*
- void **destroyMarshalers** ()  
*Cleans up all registered Marshallers and empties the dataMarshallers vector.*

## Static Protected Attributes

- static const unsigned char **NULL\_TYPE**
- static const int **DEFAULT\_VERSION** = 1

## 6.598.1 Constructor & Destructor Documentation

### 6.598.1.1 `activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat (const decaf::util::Properties & properties)`

Constructs a new **OpenWireFormat** (p. 2887) object.

#### Parameters:

*properties* - can contain optional config params.

### 6.598.1.2 `virtual activemq::wireformat::openwire::OpenWireFormat::~~OpenWireFormat ()` [virtual]

## 6.598.2 Member Function Documentation

### 6.598.2.1 `void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (marshal::DataStreamMarshaller * marshaller)`

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

#### Parameters:

*marshaller* - the Marshaler to add to the collection.

### 6.598.2.2 `virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator (const Pointer< transport::Transport > & transport) throw ( decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

#### Returns:

new instance of a **WireFormatNegotiator** (p. 4016).

Implements `activemq::wireformat::WireFormat` (p. 3977).

### 6.598.2.3 `void activemq::wireformat::openwire::OpenWireFormat::destroyMarshallers ()` [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector. This should be called before a reconfiguration of the version marshallers, or on destruction of this object

**6.598.2.4** `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * dis) throw ( decaf::io::IOException )`  
[protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object. This method can return null if the type of the object to unmarshal is NULL, empty data.

**Parameters:**

*dis* - DataInputStream to read from

**Returns:**

new DataStructure\* that the caller owns

**Exceptions:**

*IOException* if an error occurs during the unmarshal

**6.598.2.5** `int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ()`  
`const` [inline]

Returns the currently set Cache size.

**Returns:**

the current value of the broker's cache size.

**6.598.2.6** `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration ()`  
`const` [inline]

Gets the MaxInactivityDuration setting.

**Returns:**

maximum inactivity duration value in milliseconds.

**6.598.2.7** `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay ()`  
`const` [inline]

Gets the MaxInactivityDurationInitialDelay setting.

**Returns:**

maximum inactivity duration initial delay value in milliseconds.

**6.598.2.8** `virtual const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo () const [inline, virtual]`

Gets the Preferred WireFormatInfo object that this class holds.

**Returns:**

pointer to a preferred WireFormatInfo object

**6.598.2.9** `int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]`

Get the current Wireformat Version.

**Returns:**

int that identifies the version

Implements `activemq::wireformat::WireFormat` (p. 3977).

**6.598.2.10** `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this `WireFormat` (p. 3976) has a Negotiator that needs to wrap the Transport that uses it.

**Returns:**

true if the `WireFormat` (p. 3976) provides a Negotiator.

Implements `activemq::wireformat::WireFormat` (p. 3977).

**6.598.2.11** `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

**Returns:**

true while in the doUnmarshal method.

Implements `activemq::wireformat::WireFormat` (p. 3978).

**6.598.2.12** `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]`

Checks if the cacheEnabled flag is on.

**Returns:**

true if the flag is on.

**6.598.2.13** `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]`

Checks if the sizePrefixDisabled flag is on.

**Returns:**

true if the flag is on.

**6.598.2.14** `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]`

Checks if the stackTraceEnabled flag is on.

**Returns:**

true if the flag is on.

**6.598.2.15** `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

**Returns:**

true if the flag is on.

**6.598.2.16** `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]`

Checks if the tightEncodingEnabled flag is on.

**Returns:**

true if the flag is on.

**6.598.2.17** `void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * o, decaf::io::DataOutputStream * dataOut) throw ( decaf::io::IOException )`

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface. The marshaled data is written to the passed in DataOutputStream.

**Parameters:**

*o* - DataStructure derived Object to Marshal

*dataOut* - DataOutputStream to write the data to

**Exceptions:**

*IOException* if an error occurs.

**6.598.2.18** `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * dis) throw ( decaf::io::IOException )`

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format. Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

**Parameters:**

*dis* - the DataInputStream to read the data from

**Returns:**

a new DataStructure derived Object pointer

**Exceptions:**

*IOException* if an error occurs.

**6.598.2.19** `virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw ( decaf::io::IOException )`  
[virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

**Parameters:**

*command* The Command to Marshal.

*transport* (p. 97) The Transport instance that called this method.

*out* The output stream to write the command to.

**Exceptions:**

*IOException*

Implements `activemq::wireformat::WireFormat` (p. 3978).

**6.598.2.20** `void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & info) throw ( decaf::lang::exceptions::IllegalStateException )`

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

**Parameters:**

*info* - The new Wireformat Info settings

**Exceptions:**

*IllegalStateException* is wire format can't be negotiated.

**6.598.2.21** void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool *cacheEnabled*) [inline]

Sets if the cacheEnabled flag is on.

**Parameters:**

*cacheEnabled* - true to turn flag is on

**6.598.2.22** void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (int *value*) [inline]

Sets the current Cache size.

**Parameters:**

*value* - the value to send as the broker's cache size.

**6.598.2.23** void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long *value*) [inline]

Sets the MaxInactivityDuration setting.

**Parameters:**

*value* - the Max inactivity duration value in milliseconds.

**6.598.2.24** void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay (long long *value*) [inline]

Sets the MaxInactivityDurationInitialDelay setting.

**Parameters:**

*value* - the Max inactivity Initial Delay duration value in milliseconds.

**6.598.2.25** virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo (const Pointer< commands::WireFormatInfo > & *info*) throw ( decaf::lang::exceptions::IllegalStateException ) [virtual]

Configures this object using the provided WireformatInfo object.

**Parameters:**

*info* - a WireFormatInfo object, takes ownership.

**6.598.2.26** void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

**Parameters:**

*sizePrefixDisabled* - true to turn flag is on

**6.598.2.27** void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

**Parameters:**

*stackTraceEnabled* - true to turn flag is on

**6.598.2.28** void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

**Parameters:**

*tcpNoDelayEnabled* - true to turn flag is on

**6.598.2.29** void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*) [inline]

Sets if the tightEncodingEnabled flag is on.

**Parameters:**

*tightEncodingEnabled* - true to turn flag is on

**6.598.2.30** void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Set the current Wireformat Version.



**Parameters:**

*version* - int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3978).

**6.598.2.31** `virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1(commands::DataStructure * object, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Utility method for Tight Marshaling the given object to the boolean stream passed.

**Parameters:**

*object* - The DataStructure to **marshal** (p. 107)

*bs* - the BooleanStream to write to

**Returns:**

size of the data returned.

**6.598.2.32** `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2(commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method that will Tight **marshal** (p. 107) some internally nested object that implements the DataStructure interface. Writes the data to the Data Output Stream provided.

**Parameters:**

*o* - DataStructure object

*ds* - DataOutputStream for writing

*bs* - BooleanStream

**Exceptions:**

*IOException* if an error occurs.

**6.598.2.33** `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject(decaf::io::DataInputStream * dis, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream. The DataStructure instance that is returned is now the property of the caller.

**Parameters:**

*dis* - DataInputStream to read from

*bs* - BooleanStream to read from

**Returns:**

Newly allocated DataStructure Object

**Exceptions:**

*IOException* if an error occurs.

**6.598.2.34** `virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * transport, decaf::io::DataInputStream * in) throw ( decaf::io::IOException ) [virtual]`

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form. Returns a Pointer to the newly un-marshaled Command.

**Parameters:**

*transport* (p. 97) - Pointer to the **transport** (p. 97) that is making this request.

*in* - the input stream to read the command from.

**Returns:**

the newly marshaled Command, caller owns the pointer

**Exceptions:**

*IOException*

Implements **activemq::wireformat::WireFormat** (p. 3979).

## 6.598.3 Field Documentation

**6.598.3.1** `const int activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION = 1 [static, protected]`

**6.598.3.2** `const unsigned char activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE [static, protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormat.h`

## 6.599 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatFactory:

### Public Member Functions

- **OpenWireFormatFactory** ()

*Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.*

- virtual **~OpenWireFormatFactory** ()

- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::IllegalStateException** )

*Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.599.1 Constructor & Destructor Documentation

#### 6.599.1.1 activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory () [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called. URL options ----- wireFormat.stackTraceEnabled wireFormat.cacheEnabled wireFormat.tcpNoDelayEnabled wireFormat.tightEncodingEnabled wireFormat.sizePrefixDisabled wireFormat.maxInactivityDuration wireFormat.maxInactivityDurationInitialDelay

#### 6.599.1.2 virtual activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory () [inline, virtual]

### 6.599.2 Member Function Documentation

#### 6.599.2.1 virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const **decaf::util::Properties** & *properties*) throw ( **decaf::lang::exceptions::IllegalStateException** ) [virtual]

Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters:

*properties* - the Properties for this **WireFormat** (p. 3976)

Implements **activemq::wireformat::WireFormatFactory** (p. 3980).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

## 6.600 activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

### Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** \*wireFormat, const **Pointer**< **transport::Transport** > &next)  
*Constructor - Initializes this object around another Transport.*
- virtual **~OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **onTransportException** (**transport::Transport** \*source, const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*

### 6.600.1 Constructor & Destructor Documentation

#### 6.600.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator (**OpenWireFormat** \* wireFormat, const **Pointer**< **transport::Transport** > & next)

Constructor - Initializes this object around another Transport.

**Parameters:**

*wireFormat* - The **WireFormat** (p. 3976) object we use to negotiate

*next* - The next **transport** (p. 97) in the chain

**6.600.1.2**    **virtual**  
                  **activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator**  
                  **()**    [virtual]

**6.600.2 Member Function Documentation**

**6.600.2.1**    **virtual void ac-**  
                  **tivemq::wireformat::openwire::OpenWireFormatNegotiator::close ()**  
                  **throw ( decaf::io::IOException )**    [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions:**

*IOException* if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3893).

**6.600.2.2**    **virtual void ac-**  
                  **tivemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand**  
                  **(const Pointer< commands::Command > & command)**    [virtual]

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

**Parameters:**

*command* the received from the nested **transport** (p. 97).

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

**6.600.2.3**    **virtual void ac-**  
                  **tivemq::wireformat::openwire::OpenWireFormatNegotiator::oneway**  
                  **(const Pointer< commands::Command > &**  
                  **command) throw ( decaf::io::IOException, de-**  
                  **cafe::lang::exceptions::UnsupportedOperationException )**  
                  [virtual]

Sends a one-way command. Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

**Parameters:**

*command* the command to be sent.

**Exceptions:**

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

**6.600.2.4** `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command **transport** (p. 97).

**Parameters:**

*source* The source of the exception  
*ex* The exception.

**6.600.2.5** `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

**Parameters:**

*command* The request to send.  
*timeout* The time to wait for the response.

**Returns:**

the response from the server.

**Exceptions:**

*IOException* if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3896).

**6.600.2.6** `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

**Parameters:**

*command* The request to send.

**Returns:**

the response from the server.

**Exceptions:**

***IOException*** if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3897).

**6.600.2.7** **virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start ()**  
**throw ( decaf::io::IOException ) [virtual]**

Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions:**

***IOException*** if an error occurs or if this **transport** (p. 97) has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3898).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h`



## 6.601 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>Inheritance diagram for activemq::wireformat::openwire::OpenWireResponseBuilder:

### Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command > &command**)

*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*

- virtual void **buildIncomingCommands** (const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue**)

*When called the ResponseBuilder must construct all the Responses or Asynchronous **commands** (p. 87) that would be sent to this client by the Broker upon receipt of the passed command.*

### 6.601.1 Constructor & Destructor Documentation

**6.601.1.1** **activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder** () [inline]

**6.601.1.2** **virtual**  
**activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder** () [inline, virtual]

### 6.601.2 Member Function Documentation

**6.601.2.1** **virtual void** **activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands** (const **Pointer< commands::Command > & command**, **decaf::util::StlQueue< Pointer< commands::Command > > & queue**) [virtual]

When called the ResponseBuilder must construct all the Responses or Asynchronous **commands** (p. 87) that would be sent to this client by the Broker upon receipt of the passed command.

#### Parameters:

*command* - The Command being sent to the Broker.

*queue* - Queue of Command sent back from the broker.

Implements **activemq::transport::mock::ResponseBuilder** (p. 3289).

**6.601.2.2** `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse(const Pointer< commands::Command > & command) [virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

**Parameters:**

*command* - The command to build a response for

**Returns:**

A Response object pointer, or NULL if no response.

Implements `activemq::transport::mock::ResponseBuilder` (p. 3290).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h`

## 6.602 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

#include <src/main/decaf/io/OutputStream.h> Inheritance diagram for decaf::io::OutputStream:

### Public Member Functions

- **OutputStream** ()
- virtual **~OutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*  
**Exceptions:**  
*IOException* ( p. 2142) if an error occurs while closing.
- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*  
**Exceptions:**  
*IOException* ( p. 2142) if an I/O error occurs.
- virtual void **write** (unsigned char c) throw ( decaf::io::IOException )  
*Writes a single byte to the output stream.*
- virtual void **write** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream.*
- virtual void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0 throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.602.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

**Since:**

1.0

## 6.602.2 Constructor & Destructor Documentation

**6.602.2.1** decaf::io::OutputStream::OutputStream ()

**6.602.2.2** virtual decaf::io::OutputStream::~~OutputStream () [virtual]

## 6.602.3 Member Function Documentation

**6.602.3.1** virtual void decaf::io::OutputStream::close () throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing.

Implements decaf::io::Closeable (p. 1149).

Reimplemented in decaf::internal::io::StandardErrorOutputStream (p. 3580), decaf::internal::io::StandardOutputStream (p. 3584), decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream (p. 2886), decaf::internal::net::tcp::TcpSocketOutputStream (p. 3751), decaf::io::FilterOutputStream (p. 1901), and decaf::util::zip::DeflaterOutputStream (p. 1718).

**6.602.3.2** virtual void decaf::io::OutputStream::doWriteArray (const unsigned char \* *buffer*, int *size*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented in decaf::io::BufferedOutputStream (p. 941), and decaf::io::FilterOutputStream (p. 1902).

**6.602.3.3** virtual void decaf::io::OutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented in activemq::io::LoggingOutputStream (p. 2401), decaf::internal::io::StandardErrorOutputStream (p. 3580), decaf::internal::io::StandardOutputStream (p. 3585), decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream (p. 2886), decaf::internal::net::tcp::TcpSocketOutputStream (p. 3751), decaf::io::BufferedOutputStream (p. 941), decaf::io::ByteArrayOutputStream (p. 1029), decaf::io::DataOutputStream (p. 1580), decaf::io::FilterOutputStream (p. 1902), decaf::util::zip::CheckedOutputStream (p. 1141), and decaf::util::zip::DeflaterOutputStream (p. 1719).

**6.602.3.4** virtual void decaf::io::OutputStream::doWriteByte (unsigned char *value*)  
throw ( decaf::io::IOException ) [protected, pure virtual]

Implemented in **activemq::io::LoggingOutputStream** (p. 2402),  
**decaf::internal::io::StandardErrorOutputStream** (p. 3580),  
**decaf::internal::io::StandardOutputStream** (p. 3585), **de-**  
**caf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2886),  
**decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3751), **de-**  
**caf::io::BufferedOutputStream** (p. 941), **decaf::io::ByteArrayOutputStream**  
(p. 1029), **decaf::io::DataOutputStream** (p. 1581), **decaf::io::FilterOutputStream**  
(p. 1902), **decaf::util::zip::CheckedOutputStream** (p. 1141), and **de-**  
**caf::util::zip::DeflaterOutputStream** (p. 1719).

**6.602.3.5** virtual void decaf::io::OutputStream::flush () throw (  
decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

#### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

The default implementation of this method does nothing.

Implements **decaf::io::Flushable** (p. 1936).

Reimplemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3580), **de-**  
**caf::internal::io::StandardOutputStream** (p. 3585), **decaf::io::BufferedOutputStream**  
(p. 941), and **decaf::io::FilterOutputStream** (p. 1902).

**6.602.3.6** virtual void decaf::io::OutputStream::lock () throw (  
decaf::lang::exceptions::RuntimeException ) [inline, virtual]

Locks the object.

#### Exceptions:

**RuntimeException** if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3701).

**6.602.3.7** virtual void decaf::io::OutputStream::notify ()  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

**IllegalMonitorStateException** - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3702).

**6.602.3.8** `virtual void decaf::io::OutputStream::notifyAll ()  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.602.3.9** `virtual std::string decaf::io::OutputStream::toString () const [virtual]`

Output a String representation of this object. The default version of this method just prints the Class Name.

#### Returns:

a string representation of the object.

Reimplemented in **decaf::io::ByteArrayOutputStream** (p. 1030), and **decaf::io::FilterOutputStream** (p. 1903).

**6.602.3.10** `virtual bool decaf::io::OutputStream::tryLock () throw (  
decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns:

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions:

*RuntimeException* if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

**6.602.3.11** `virtual void decaf::io::OutputStream::unlock () throw (  
decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).

**6.602.3.12** virtual void decaf::io::OutputStream::wait (long long *millisecs*, int *nanos*) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

**Exceptions:**

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.602.3.13** virtual void decaf::io::OutputStream::wait (long long *millisecs*) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.



***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

**6.602.3.14** `virtual void decaf::io::OutputStream::wait ()  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

**6.602.3.15** `virtual void decaf::io::OutputStream::write (const unsigned  
char * buffer, int size, int offset, int length) throw (  
decaf::io::IOException, decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Writes an array of bytes to the output stream in order starting at `buffer[offset]` and proceeding until the number of bytes specified by the `length` argument are written or an error occurs. The default implementation of this method simply calls the `doWriteArrayBounded` method which writes the contents of the array using the `doWriteByte` method repeatedly. It is recommended that a subclass override `doWriteArrayBounded` to provide more performant means of writing the array.

#### Parameters:

***buffer*** The array of bytes to write.

***size*** The size of the buffer array passed.

***offset*** The position to start writing in buffer.

***length*** The number of bytes from the buffer to be written.

#### Exceptions:

***IOException*** (p. 2142) if an I/O error occurs.

***NullPointerException*** thrown if buffer is Null.

***IndexOutOfBoundsException*** if the `offset + length > size`. or one of the parameters is negative.

**6.602.3.16** `virtual void decaf::io::OutputStream::write (const unsigned char * buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Writes an array of bytes to the output stream. The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the `doWriteArray` which writes the contents of the array using the `doWriteByte` method repeatedly. It is recommended that a subclass override `doWriteArray` to provide more performant means of writing the array.

**Parameters:**

*buffer* The vector of bytes to write.  
*size* The size of the buffer passed.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.  
*NullPointerException* thrown if buffer is Null.  
*IndexOutOfBoundsException* if size value is negative.

**6.602.3.17** `virtual void decaf::io::OutputStream::write (unsigned char c) throw ( decaf::io::IOException ) [virtual]`

Writes a single byte to the output stream. The default implementation of this method calls the pure virtual method `doWriteByte` which must be implemented by any subclass of the **OutputStream** (p. 2907).

**Parameters:**

*c* The byte to write to the sink.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/OutputStream.h`

## 6.603 decaf::io::OutputStreamWriter Class Reference

A class for turning a character stream into a byte stream.

#include <src/main/decaf/io/OutputStreamWriter.h> Inheritance diagram for decaf::io::OutputStreamWriter:

### Public Member Functions

- **OutputStreamWriter** (**OutputStream** \*stream, bool own=false)  
*Creates a new **OutputStreamWriter** (p. 2915).*
- virtual ~**OutputStreamWriter** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

### Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).*
- virtual void **checkClosed** () const throw ( decaf::io::IOException )

### 6.603.1 Detailed Description

A class for turning a character stream into a byte stream.

See also:

**InputStreamReader** (p. 2053)

Since:

1.0

### 6.603.2 Constructor & Destructor Documentation

#### 6.603.2.1 decaf::io::OutputStreamWriter::OutputStreamWriter (**OutputStream** \*stream, bool own = false)

Creates a new **OutputStreamWriter** (p. 2915).

**Parameters:**

- stream* The **OutputStream** (p. 2907) to wrap. (cannot be NULL).
- own* Indicates whether this instance own the given **OutputStream** (p. 2907). If true then the **OutputStream** (p. 2907) is destroyed when this class is.

**Exceptions:**

- NullPointerException* if the stream is NULL.

**6.603.2.2** virtual decaf::io::OutputStreamWriter::~~OutputStreamWriter ()  
[virtual]

**6.603.3 Member Function Documentation**

**6.603.3.1** virtual void decaf::io::OutputStreamWriter::checkClosed () const throw (  
decaf::io::IOException ) [protected, virtual]

**6.603.3.2** virtual void decaf::io::OutputStreamWriter::close () throw (  
decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

**Exceptions:**

- IOException* (p. 2142) if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 1149).

**6.603.3.3** virtual void decaf::io::OutputStreamWriter::doWriteArrayBounded  
(const char \* *buffer*, int *size*, int *offset*, int *length*) throw (  
decaf::io::IOException, decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [protected,  
virtual]

Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ). All subclasses must override this method to provide the basic **Writer** (p. 4021) functionality.

Implements **decaf::io::Writer** (p. 4024).

**6.603.3.4** virtual void decaf::io::OutputStreamWriter::flush () throw (  
decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

**Exceptions:**

- IOException* (p. 2142) if an I/O error occurs.

Implements **decaf::io::Flushable** (p. 1936).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/OutputStreamWriter.h`

## 6.604 activemq::commands::PartialCommand Class Reference

#include <src/main/activemq/commands/PartialCommand.h> Inheritance diagram for activemq::commands::PartialCommand:

### Public Member Functions

- **PartialCommand** ()
- virtual **~PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **PartialCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

### Static Public Attributes

- static const unsigned char **ID\_PARTIALCOMMAND** = 60

### Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

## 6.604.1 Constructor & Destructor Documentation

**6.604.1.1** `activemq::commands::PartialCommand::PartialCommand ()`

**6.604.1.2** `virtual activemq::commands::PartialCommand::~~PartialCommand ()`  
[virtual]

## 6.604.2 Member Function Documentation

**6.604.2.1** `virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2301).

**6.604.2.2** `virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2302).

**6.604.2.3** `virtual bool activemq::commands::PartialCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented in `activemq::commands::LastPartialCommand` (p. 2302).

**6.604.2.4** `virtual int activemq::commands::PartialCommand::getCommandId ()`  
`const [virtual]`

**6.604.2.5** `virtual std::vector<unsigned char>& ac-`  
`tivemq::commands::PartialCommand::getData ()`  
`[virtual]`

**6.604.2.6** `virtual const std::vector<unsigned char>& ac-`  
`tivemq::commands::PartialCommand::getData () const`  
`[virtual]`

**6.604.2.7** `virtual unsigned char ac-`  
`tivemq::commands::PartialCommand::getDataStructureType () const`  
`[virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2302).

**6.604.2.8** `virtual void activemq::commands::PartialCommand::setCommandId (int`  
`commandId) [virtual]`

**6.604.2.9** `virtual void activemq::commands::PartialCommand::setData (const`  
`std::vector< unsigned char > & data) [virtual]`

**6.604.2.10** `virtual std::string activemq::commands::PartialCommand::toString ()`  
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 833).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2302).



### 6.604.3 Field Documentation

**6.604.3.1** `int activemq::commands::PartialCommand::commandId` [protected]

**6.604.3.2** `std::vector<unsigned char> activemq::commands::PartialCommand::data`  
[protected]

**6.604.3.3** `const unsigned char activemq::commands::PartialCommand::ID_ -  
PARTIALCOMMAND = 60` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

## 6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2922).

#include <src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.605.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2922).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.605.2 Constructor & Destructor Documentation

**6.605.2.1** `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

**6.605.2.2** `virtual activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

## 6.605.3 Member Function Documentation

**6.605.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2305).

**6.605.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2305).

**6.605.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2305).

**6.605.3.4 virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2306).

**6.605.3.5 virtual int activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2306).

**6.605.3.6** virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2307).

**6.605.3.7** virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2307).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h

## 6.606 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2926).

#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.606.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2926).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.606.2 Constructor & Destructor Documentation

**6.606.2.1** `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller()` `[inline]`

**6.606.2.2** `virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller()` `[inline, virtual]`

## 6.606.3 Member Function Documentation

**6.606.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2313).

**6.606.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2313).

**6.606.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2313).

**6.606.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2314).

**6.606.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2314).



**6.606.3.6** virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2315).

**6.606.3.7** virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2315).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**

## 6.607 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2930).

#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.607.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2930).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.607.2 Constructor & Destructor Documentation

**6.607.2.1** `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

**6.607.2.2** `virtual activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

## 6.607.3 Member Function Documentation

**6.607.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2317).

**6.607.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2317).

**6.607.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2317).

**6.607.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2318).

**6.607.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2318).

**6.607.3.6** virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2319).

**6.607.3.7** virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h

## 6.608 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2934).

#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.608.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2934).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.608.2 Constructor & Destructor Documentation

**6.608.2.1** `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

**6.608.2.2** `virtual activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

## 6.608.3 Member Function Documentation

**6.608.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2309).

**6.608.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2309).

**6.608.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2309).

**6.608.3.4 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2310).

**6.608.3.5 virtual int activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2310).



**6.608.3.6** virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2311).

**6.608.3.7** virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**PartialCommandMarshaller.h**

## 6.609 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2938).

#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.609.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2938).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.609.2 Constructor & Destructor Documentation

**6.609.2.1** `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

**6.609.2.2** `virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

## 6.609.3 Member Function Documentation

**6.609.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2321).

**6.609.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2321).

**6.609.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2321).

**6.609.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2322).

**6.609.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2322).

**6.609.3.6** virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2323).

**6.609.3.7** virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2323).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**PartialCommandMarshaller.h**

## 6.610 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2942).

#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.610.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2942).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.610.2 Constructor & Destructor Documentation

**6.610.2.1** `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller()` `[inline]`

**6.610.2.2** `virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller()` `[inline, virtual]`

## 6.610.3 Member Function Documentation

**6.610.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2325).

**6.610.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2325).

**6.610.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2325).

**6.610.3.4 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2326).

**6.610.3.5 virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2326).



**6.610.3.6** virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2327).

**6.610.3.7** virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2327).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**PartialCommandMarshaller.h**

## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

### Public Types

- typedef T \* **PointerType**
- typedef T & **ReferenceType**
- typedef REFCOUNTER **CounterType**

### Public Member Functions

- **Pointer** ()  
*Default Constructor.*
- **Pointer** (const **PointerType** value)  
*Explicit Constructor, creates a **Pointer** (p. 2946) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()  
*Copy constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value) throw ()  
*Copy constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value, const **STATIC\_CAST\_TOKEN** &) throw ()  
*Static Cast constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value, const **DYNAMIC\_CAST\_TOKEN** &) throw ( decaf::lang::exceptions::ClassCastException )  
*Dynamic Cast constructor.*
- virtual ~**Pointer** () throw ()
- void **reset** (T \*value)  
*Resets the **Pointer** (p. 2946) to hold the new value.*
- T \* **release** ()  
*Releases the **Pointer** (p. 2946) held and resets the **internal** (p. 144) pointer value to Null.*
- **PointerType** **get** () const  
*Gets the real pointer that is contained within this **Pointer** (p. 2946).*
- void **swap** (**Pointer** &value) throw ()

*Exception (p. 1831) Safe Swap Function.*

- **Pointer** & **operator**== (const **Pointer** &right) throw ()  
*Assigns the value of right to this **Pointer** (p. 2946) and increments the reference Count.*
- template<typename T1 , typename R1 >  
**Pointer** & **operator**== (const **Pointer**< T1, R1 > &right) throw ()
- **ReferenceType operator\*** ()  
*Dereference Operator, returns a reference to the Contained value.*
- **ReferenceType operator\*** () const
- **PointerType operator->** ()  
*Indirection Operator, returns a pointer to the Contained value.*
- **PointerType operator->** () const
- bool **operator!** () const
- template<typename T1 , typename R1 >  
 bool **operator**== (const **Pointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >  
 bool **operator**!= (const **Pointer**< T1, R1 > &right) const
- template<typename T1 >  
**Pointer**< T1, **CounterType** > **dynamicCast** () const
- template<typename T1 >  
**Pointer**< T1, **CounterType** > **staticCast** () const

## Friends

- bool **operator**== (const **Pointer** &left, const T \*right)
- bool **operator**== (const T \*left, const **Pointer** &right)
- bool **operator**!= (const **Pointer** &left, const T \*right)
- bool **operator**!= (const T \*left, const **Pointer** &right)

### 6.611.1 Detailed Description

```
template<typename          T,          typename          REFCOUNTER          =
decaf::util::concurrent::atomic::AtomicRefCounter>      class      decaf::lang::Pointer<
T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used. This **Pointer** (p. 2946) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2946) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2946) in a STL container that requires it, **Pointer** (p. 2946) provides an implementation of std::less.

Since:

1.0

## 6.611.2 Member Typedef Documentation

- 6.611.2.1** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType`
- 6.611.2.2** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType`
- 6.611.2.3** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

## 6.611.3 Constructor & Destructor Documentation

- 6.611.3.1** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer () [inline]`

Default Constructor. Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::Pointer< TransactionId >::reset()`.

- 6.611.3.2** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType value) [inline, explicit]`

Explicit Constructor, creates a **Pointer** (p. 2946) that contains value with a single reference. This object now has ownership until a call to release.

### Parameters:

*value* - instance of the type we are containing here.

- 6.611.3.3** `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T, REFCOUNTER > & value) throw () [inline]`

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

**6.611.3.4** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer  
(const Pointer< T1, R1 > & value) throw () [inline]`

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

**6.611.3.5** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer  
(const Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN &)  
throw () [inline]`

Static Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p. 2946) object.

**Parameters:**

*value* - **Pointer** (p. 2946) instance to cast to this type.

**6.611.3.6** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer  
(const Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN  
&) throw ( decaf::lang::exceptions::ClassCastException ) [inline]`

Dynamic Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p. 2946) object. If the cast fails and return NULL then this method throws a **ClassCastException**.

**Parameters:**

*value* - **Pointer** (p. 2946) instance to cast to this type.

**Exceptions:**

*ClassCastException* if the dynamic cast returns NULL

**6.611.3.7** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> virtual  
decaf::lang::Pointer< T, REFCOUNTER >::~~Pointer () throw ()  
[inline, virtual]`

#### 6.611.4 Member Function Documentation

**6.611.4.1** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T,  
REFCOUNTER >::dynamicCast () const [inline]`

**6.611.4.2** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> PointerType  
decaf::lang::Pointer< T, REFCOUNTER >::get () const [inline]`

Gets the real pointer that is contained within this **Pointer** (p. 2946). This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2946). Use at your own risk.

##### Returns:

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::state::ConnectionState::getTransactionState()`, `decaf::lang::operator!=()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `std::less< decaf::lang::Pointer< T > >::operator()`, `decaf::lang::operator==()`, `decaf::lang::Pointer< TransactionId >::operator==()`, and `activemq::state::ConnectionState::removeTempDestination()`.

**6.611.4.3** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> bool  
decaf::lang::Pointer< T, REFCOUNTER >::operator! () const [inline]`

**6.611.4.4** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename  
T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER  
>::operator!= (const Pointer< T1, R1 > & right) const [inline]`

**6.611.4.5** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType  
decaf::lang::Pointer< T, REFCOUNTER >::operator* () const [inline]`

**6.611.4.6** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType  
decaf::lang::Pointer< T, REFCOUNTER >::operator* () [inline]`

Dereference Operator, returns a reference to the Contained value. This method throws an `NullPointerException` if the contained value is `NULL`.

##### Returns:

reference to the contained pointer.

**Exceptions:**

*NullPointerException* if the contained value is Null

**6.611.4.7** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
decaf::lang::Pointer< T, REFCOUNTER >::operator-> () const [inline]`

**6.611.4.8** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
decaf::lang::Pointer< T, REFCOUNTER >::operator-> () [inline]`

Indirection Operator, returns a pointer to the Contained value. This method throws an *NullPointerException* if the contained value is NULL.

**Returns:**

reference to the contained pointer.

**Exceptions:**

*NullPointerException* if the contained value is Null

**6.611.4.9** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename  
T1 , typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER  
>::operator= (const Pointer< T1, R1 > & right) throw () [inline]`

**6.611.4.10** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> Pointer&  
decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer<  
T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 2946) and increments the reference Count.

**Parameters:**

*right* - **Pointer** (p. 2946) on the right hand side of an operator= call to this.

**6.611.4.11** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename  
T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER  
>::operator== (const Pointer< T1, R1 > & right) const [inline]`

**6.611.4.12** `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> T*  
decaf::lang::Pointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p. 2946) held and resets the **internal** (p. 144) pointer value to Null. This method is not guaranteed to be safe if the **Pointer** (p. 2946) is held by more than one object or this method is called from more than one thread.

**Parameters:**

*value* - The new value to contain.

**Returns:**

The pointer instance that was held by this **Pointer** (p. 2946) object, the pointer is no longer owned by this **Pointer** (p. 2946) and won't be freed when this **Pointer** (p. 2946) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

```
6.611.4.13  template<typename T, typename REFCOUNTER =
              decaf::util::concurrent::atomic::AtomicRefCounter> void
              decaf::lang::Pointer< T, REFCOUNTER >::reset (T * value) [inline]
```

Resets the **Pointer** (p. 2946) to hold the new value. Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2946) to a NULL pointer.

**Parameters:**

*value* - The new value to contain.

```
6.611.4.14  template<typename T, typename REFCOUNTER =
              decaf::util::concurrent::atomic::AtomicRefCounter> template<typename
              T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T,
              REFCOUNTER >::staticCast () const [inline]
```

```
6.611.4.15  template<typename T, typename REFCOUNTER =
              decaf::util::concurrent::atomic::AtomicRefCounter> void
              decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T,
              REFCOUNTER > & value) throw () [inline]
```

**Exception** (p. 1831) Safe Swap Function.

**Parameters:**

*value* - the value to swap with this.

Referenced by `decaf::lang::Pointer< TransactionId >::operator=()`, and `decaf::lang::Pointer< TransactionId >::swap()`.



### 6.611.5 Friends And Related Function Documentation

**6.611.5.1**    `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> bool operator!=  
(const T * left, const Pointer< T, REFCOUNTER > & right)` [friend]

**6.611.5.2**    `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> bool operator!=  
(const Pointer< T, REFCOUNTER > & left, const T * right)` [friend]

**6.611.5.3**    `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> bool operator==  
(const T * left, const Pointer< T, REFCOUNTER > & right)` [friend]

**6.611.5.4**    `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> bool operator==  
(const Pointer< T, REFCOUNTER > & left, const T * right)` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.612 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2946) instance.

#include <src/main/decaf/lang/Pointer.h> Inheritance diagram for decaf::lang::PointerComparator< T, R >:

### Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

### 6.612.1 Detailed Description

template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>  
class decaf::lang::PointerComparator< T, R >

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2946) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2946) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

### 6.612.2 Member Function Documentation

**6.612.2.1** template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual  
int decaf::lang::PointerComparator< T, R >::compare (const **Pointer**< T, R > & *left*, const **Pointer**< T, R > & *right*) const [inline, virtual]

**6.612.2.2** template<typename T , typename R = decaf::util::concurrent::atomic::AtomicRefCounter> virtual  
bool decaf::lang::PointerComparator< T, R >::operator() (const **Pointer**< T, R > & *left*, const **Pointer**< T, R > & *right*) const [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

## 6.613 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

#include <src/main/activemq/cmsutil/PooledSession.h> Inheritance diagram for activemq::cmsutil::PooledSession:

### Public Member Functions

- **PooledSession** (SessionPool \*pool, cms::Session \*session)
- virtual ~**PooledSession** ()  
*Does nothing.*
- virtual cms::Session \* **getSession** ()  
*Returns a non-constant reference to the internal session object.*
- virtual const cms::Session \* **getSession** () const  
*Returns a constant reference to the internal session object.*
- virtual void **close** () throw ( cms::CMSEException )  
*Returns this session back to the pool, but does not close or destroy the internal session object.*
- virtual void **commit** () throw ( cms::CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** () throw ( cms::CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** () throw ( cms::CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual cms::MessageConsumer \* **createConsumer** (const cms::Destination \*destination) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination.*
- virtual cms::MessageConsumer \* **createConsumer** (const cms::Destination \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual cms::MessageConsumer \* **createConsumer** (const cms::Destination \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual cms::MessageConsumer \* **createDurableConsumer** (const cms::Topic \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSEException )  
*Creates a durable subscriber to the specified topic, using a Message selector.*

- virtual **cms::MessageConsumer** \* **createCachedConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )  
*First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.*
- virtual **cms::MessageProducer** \* **createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*Creates a MessageProducer to send messages to the specified destination.*
- virtual **cms::MessageProducer** \* **createCachedProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*First checks the internal producer cache and creates one if none exist for the given destination.*
- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSEException )  
*Creates a new QueueBrowser to peek at Messages on the given Queue.*
- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSEException )  
*Creates a new QueueBrowser to peek at Messages on the given Queue.*
- virtual **cms::Queue** \* **createQueue** (const std::string &queueName) throw ( cms::CMSEException )  
*Creates a queue identity given a Queue name.*
- virtual **cms::Topic** \* **createTopic** (const std::string &topicName) throw ( cms::CMSEException )  
*Creates a topic identity given a Queue name.*
- virtual **cms::TemporaryQueue** \* **createTemporaryQueue** () throw ( cms::CMSEException )  
*Creates a TemporaryQueue object.*
- virtual **cms::TemporaryTopic** \* **createTemporaryTopic** () throw ( cms::CMSEException )  
*Creates a TemporaryTopic object.*
- virtual **cms::Message** \* **createMessage** () throw ( cms::CMSEException )  
*Creates a new Message.*
- virtual **cms::BytesMessage** \* **createBytesMessage** () throw ( cms::CMSEException )  
*Creates a BytesMessage.*
- virtual **cms::BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSEException )  
*Creates a BytesMessage and sets the payload to the passed value.*
- virtual **cms::StreamMessage** \* **createStreamMessage** () throw ( cms::CMSEException )  
*Creates a new StreamMessage.*

- virtual **cms::TextMessage** \* **createTextMessage** () throw ( cms::CMSEException )  
*Creates a new TextMessage.*
- virtual **cms::TextMessage** \* **createTextMessage** (const std::string &text) throw ( cms::CMSEException )  
*Creates a new TextMessage and set the text to the value given.*
- virtual **cms::MapMessage** \* **createMapMessage** () throw ( cms::CMSEException )  
*Creates a new MapMessage.*
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw ( cms::CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const throw ( cms::CMSEException )  
*Gets if the Sessions is a Transacted Session.*
- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*

## Protected Member Functions

- **PooledSession** (const **PooledSession** &)
- **PooledSession** & **operator=** (const **PooledSession** &)

### 6.613.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

### 6.613.2 Constructor & Destructor Documentation

- 6.613.2.1** **activemq::cmsutil::PooledSession::PooledSession** (const **PooledSession** &)  
[inline, protected]
- 6.613.2.2** **activemq::cmsutil::PooledSession::PooledSession** (SessionPool \* *pool*, cms::Session \* *session*)
- 6.613.2.3** **virtual activemq::cmsutil::PooledSession::~~PooledSession** () [virtual]

Does nothing.

### 6.613.3 Member Function Documentation

- 6.613.3.1** **virtual void activemq::cmsutil::PooledSession::close** () throw ( cms::CMSEException ) [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Implements **cms::Session** (p. 3368).

**6.613.3.2** `virtual void activemq::cmsutil::PooledSession::commit () throw ( cms::CMSException ) [inline, virtual]`

Commits all messages done in this transaction and releases any locks currently held.

**Exceptions:**

*CMSException* - If an internal error occurs.

*IllegalStateException* - if the method is not called by a transacted session.

Implements **cms::Session** (p. 3369).

References `cms::Session::commit()`.

**6.613.3.3** `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw ( cms::CMSException ) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters:**

*queue* the Queue to browse

*selector* the Message selector to filter which messages are browsed.

**Returns:**

New QueueBrowser that is owned by the caller.

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if the destination given is invalid.

Implements **cms::Session** (p. 3369).

**6.613.3.4** `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue) throw ( cms::CMSException ) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters:**

*queue* the Queue to browse

**Returns:**

New QueueBrowser that is owned by the caller.

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if the destination given is invalid.

Implements **cms::Session** (p. 3369).

**6.613.3.5** `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * bytes, int bytesSize) throw ( cms::CMSException) [inline, virtual]`

Creates a BytesMessage and sets the payload to the passed value.

**Parameters:**

*bytes* an array of bytes to set in the message

*bytesSize* the size of the bytes array, or number of bytes to use

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements **cms::Session** (p. 3370).

**6.613.3.6** `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage () throw ( cms::CMSException) [inline, virtual]`

Creates a BytesMessage.

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements **cms::Session** (p. 3370).

**6.613.3.7** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw ( cms::CMSException ) [virtual]`

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal. If created, the consumer is added to the pool's lifecycle manager.

**Parameters:**

*destination* the destination to receive on

*selector* the selector to use

*noLocal* whether or not to receive messages from the same connection

**Returns:**

the consumer resource

**Exceptions:**

*cms::CMSEException* (p. 1160) if something goes wrong.

**6.613.3.8** `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * destination) throw ( cms::CMSEException ) [virtual]`

First checks the internal producer cache and creates one if none exist for the given destination. If created, the producer is added to the pool's lifecycle manager.

**Parameters:**

*destination* the destination to send on

**Returns:**

the producer resource

**Exceptions:**

*cms::CMSEException* (p. 1160) if something goes wrong.

**6.613.3.9** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw ( cms::CMSEException ) [inline, virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters:**

*destination* the Destination that this consumer receiving messages for.

*selector* the Message Selector to use

*noLocal* if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns:**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* - If an internal error occurs.

*InvalidDestinationException* - if an invalid destination is specified.

*InvalidSelectorException* - if the message selector is invalid.

Implements **cms::Session** (p. 3370).



**6.613.3.10** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw ( cms::CMSException )` [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters:**

*destination* the Destination that this consumer receiving messages for.

*selector* the Message Selector to use

**Returns:**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if an invalid destination is specified.

*InvalidSelectorException* - if the message selector is invalid.

Implements **cms::Session** (p. 3371).

**6.613.3.11** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination) throw ( cms::CMSException )` [inline, virtual]

Creates a MessageConsumer for the specified destination.

**Parameters:**

*destination* the Destination that this consumer receiving messages for.

**Returns:**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSException* - If an internal error occurs.

*InvalidDestinationException* - if an invalid destination is specified.

Implements **cms::Session** (p. 3371).

**6.613.3.12** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw ( cms::CMSException )` [inline, virtual]

Creates a durable subscriber to the specified topic, using a Message selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

**Parameters:**

*destination* the topic to subscribe to  
*name* The name used to identify the subscription  
*selector* the Message Selector to use  
*noLocal* if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns:**

pointer to a new durable MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* - If an internal error occurs.  
*InvalidDestinationException* - if an invalid destination is specified.  
*InvalidSelectorException* - if the message selector is invalid.

Implements **cms::Session** (p. 3372).

**6.613.3.13** `virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage () throw ( cms::CMSEException ) [inline, virtual]`

Creates a new MapMessage.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3372).

**6.613.3.14** `virtual cms::Message* activemq::cmsutil::PooledSession::createMessage () throw ( cms::CMSEException ) [inline, virtual]`

Creates a new Message.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3372).

**6.613.3.15** `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const cms::Destination * destination) throw ( cms::CMSEException ) [inline, virtual]`

Creates a MessageProducer to send messages to the specified destination.

**Parameters:**

*destination* the Destination to send on

**Returns:**

New MessageProducer that is owned by the caller.

**Exceptions:**

*CMSEException* - If an internal error occurs.

*InvalidDestinationException* - if an invalid destination is specified.

Implements **cms::Session** (p. 3373).

**6.613.3.16** `virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue  
(const std::string & queueName) throw ( cms::CMSEException )  
[inline, virtual]`

Creates a queue identity given a Queue name.

**Parameters:**

*queueName* the name of the new Queue

**Returns:**

new Queue pointer that is owned by the caller.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3373).

**6.613.3.17** `virtual cms::StreamMessage* ac-  
tivemq::cmsutil::PooledSession::createStreamMessage ()  
throw ( cms::CMSEException ) [inline, virtual]`

Creates a new StreamMessage.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3373).

**6.613.3.18** `virtual cms::TemporaryQueue* ac-  
tivemq::cmsutil::PooledSession::createTemporaryQueue ()  
throw ( cms::CMSEException ) [inline, virtual]`

Creates a TemporaryQueue object.

**Returns:**

new TemporaryQueue pointer that is owned by the caller.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3374).

**6.613.3.19** `virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic () throw ( cms::CMSException ) [inline, virtual]`

Creates a TemporaryTopic object.

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements `cms::Session` (p. 3374).

**6.613.3.20** `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const std::string & text) throw ( cms::CMSException ) [inline, virtual]`

Creates a new TextMessage and set the text to the value given.

**Parameters:**

*text* the initial text for the message

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements `cms::Session` (p. 3374).

**6.613.3.21** `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage () throw ( cms::CMSException ) [inline, virtual]`

Creates a new TextMessage.

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements `cms::Session` (p. 3374).

**6.613.3.22** `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (const std::string & topicName) throw ( cms::CMSException ) [inline, virtual]`

Creates a topic identity given a Queue name.

**Parameters:**

*topicName* the name of the new Topic

**Returns:**

new Topic pointer that is owned by the caller.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3375).

**6.613.3.23** `virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode () const throw ( cms::CMSEException ) [inline, virtual]`

Returns the acknowledgment mode of the session.

**Returns:**

the Sessions Acknowledge Mode

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3375).

**6.613.3.24** `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession () const [inline, virtual]`

Returns a constant reference to the internal session object.

**Returns:**

the session object.

**6.613.3.25** `virtual cms::Session* activemq::cmsutil::PooledSession::getSession () [inline, virtual]`

Returns a non-constant reference to the internal session object.

**Returns:**

the session object.

**6.613.3.26** `virtual bool activemq::cmsutil::PooledSession::isTransacted () const throw ( cms::CMSEException ) [inline, virtual]`

Gets if the Sessions is a Transacted Session.

**Returns:**

transacted true - false.

**Exceptions:**

*CMSEException* - If an internal error occurs.

Implements **cms::Session** (p. 3375).

**6.613.3.27** `PooledSession& activemq::cmsutil::PooledSession::operator= (const PooledSession &) [inline, protected]`

**6.613.3.28** `virtual void activemq::cmsutil::PooledSession::recover () throw ( cms::CMSException ) [inline, virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

**Exceptions:**

*CMSException* - if the CMS provider fails to stop and restart message delivery due to some internal error.

*IllegalStateException* - if the method is called by a transacted session.

Implements `cms::Session` (p. 3376).

**6.613.3.29** `virtual void activemq::cmsutil::PooledSession::rollback () throw ( cms::CMSException ) [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

**Exceptions:**

*CMSException* - If an internal error occurs.

*IllegalStateException* - if the method is not called by a transacted session.

Implements `cms::Session` (p. 3376).

**6.613.3.30** `virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & name) throw ( cms::CMSException ) [inline, virtual]`

Unsubscribes a durable subscription that has been created by a client. This method deletes the **state** (p. 95) being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

**Parameters:**

*name* The name used to identify this subscription

**Exceptions:**

*CMSException* - If an internal error occurs.

Implements **cms::Session** (p. 3376).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

## 6.614 decaf::util::concurrent::PooledThread Class Reference

#include <src/main/decaf/util/concurrent/PooledThread.h> Inheritance diagram for decaf::util::concurrent::PooledThread:

### Public Member Functions

- **PooledThread** (**ThreadPool** \*pool)

*Constructor.*

- virtual ~**PooledThread** ()
- virtual void **run** ()

*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3777) and then grabs it and calls its run method.*

- virtual void **stop** () throw ( lang::Exception )

*Stops the Thread, thread will complete its task if currently running one, and then die.*

- virtual bool **isBusy** ()

*Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.*

- virtual void **setPooledThreadListener** (**PooledThreadListener** \*listener)

*Adds a listener to this **PooledThread** (p. 2968) to be notified when this thread starts and completes a task.*

- virtual **PooledThreadListener** \* **getPooledThreadListener** ()

*Removes a listener for this **PooledThread** (p. 2968) to be notified when this thread starts and completes a task.*

### 6.614.1 Constructor & Destructor Documentation

#### 6.614.1.1 decaf::util::concurrent::PooledThread::PooledThread (**ThreadPool** \* pool)

Constructor.

#### Parameters:

*pool* the parent **ThreadPool** (p. 3777) object



**6.614.1.2** virtual decaf::util::concurrent::PooledThread::~~PooledThread ()  
[virtual]

## 6.614.2 Member Function Documentation

**6.614.2.1** virtual PooledThreadListener\* decaf::util::concurrent::PooledThread::getPooledThreadListener ()  
[inline, virtual]

Removes a listener for this PooledThread (p.2968) to be notified when this thread starts and completes a task.

### Returns:

a pointer to this thread's listener or NULL

**6.614.2.2** virtual bool decaf::util::concurrent::PooledThread::isBusy () [inline, virtual]

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.

### Returns:

true if the Thread is busy

**6.614.2.3** virtual void decaf::util::concurrent::PooledThread::run () [virtual]

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p.3777) and then grabs it and calls its run method.

Reimplemented from **decaf::lang::Thread** (p.3771).

**6.614.2.4** virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener (PooledThreadListener \* *listener*) [inline, virtual]

Adds a listener to this PooledThread (p.2968) to be notified when this thread starts and completes a task.

### Parameters:

*listener* the listener to send notifications to.

**6.614.2.5** virtual void decaf::util::concurrent::PooledThread::stop () throw (lang::Exception) [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die. Does not block.

### Exceptions:

*Exception*

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/PooledThread.h`

## 6.615 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of `ThreadPool` (p. 3777).

#include <src/main/decaf/util/concurrent/PooledThreadListener.h> Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

### Public Member Functions

- virtual `~PooledThreadListener ()`
- virtual void `onTaskStarted (PooledThread *thread)=0`  
*Called by a pooled thread when it is about to begin executing a new task.*
- virtual void `onTaskCompleted (PooledThread *thread)=0`  
*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.*
- virtual void `onTaskException (PooledThread *thread, lang::Exception &ex)=0`  
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2968) is now no longer running.*

### 6.615.1 Detailed Description

Abstract Listener Interface for users of `ThreadPool` (p. 3777). The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 3777).

Since:

1.0

### 6.615.2 Constructor & Destructor Documentation

- 6.615.2.1 virtual  
`decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener ()`  
[inline, virtual]

### 6.615.3 Member Function Documentation

- 6.615.3.1 virtual void `decaf::util::concurrent::PooledThreadListener::onTaskCompleted (PooledThread * thread)` [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

**Parameters:**

*thread* - Pointer the the Pooled Thread that is making this call.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3780).

**6.615.3.2 virtual void decaf::util::concurrent::PooledThreadListener::onTaskException**  
(PooledThread \* *thread*, lang::Exception & *ex*) [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2968) is now no longer running.

**Parameters:**

*thread* - Pointer to the Pooled Thread that is making this call

*ex* - The Exception that occurred.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3780).

**6.615.3.3 virtual void decaf::util::concurrent::PooledThreadListener::onTaskStarted**  
(PooledThread \* *thread*) [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

**Parameters:**

*thread* - Pointer to the Pooled Thread that is making this call

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3781).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThreadListener.h**

## 6.616 decaf::net::PortUnreachableException Class Reference

#include <src/main/decaf/net/PortUnreachableException.h> Inheritance diagram for decaf::net::PortUnreachableException:

### Public Member Functions

- **PortUnreachableException** () throw ()  
*Default Constructor.*
- **PortUnreachableException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()  
*Copy Constructor.*
- **PortUnreachableException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **PortUnreachableException** (const std::exception \*cause) throw ()  
*Constructor.*
- **PortUnreachableException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **PortUnreachableException** \* clone () const  
*Clones this exception.*
- virtual ~**PortUnreachableException** () throw ()

### 6.616.1 Constructor & Destructor Documentation

#### 6.616.1.1 decaf::net::PortUnreachableException::PortUnreachableException () throw () [inline]

Default Constructor.

#### 6.616.1.2 decaf::net::PortUnreachableException::PortUnreachableException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.616.1.3 decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.616.1.4 decaf::net::PortUnreachableException::PortUnreachableException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.616.1.5 decaf::net::PortUnreachableException::PortUnreachableException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.616.1.6 decaf::net::PortUnreachableException::PortUnreachableException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.616.1.7**   **virtual**  
**decaf::net::PortUnreachableException::~~PortUnreachableException ()**  
**throw ()**   [inline, virtual]

## 6.616.2 Member Function Documentation

**6.616.2.1**   **virtual PortUnreachableException\* de-**  
**caf::net::PortUnreachableException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3522).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**PortUnreachableException.h**

## 6.617 activemq::core::PrefetchPolicy Class Reference

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

#include <src/main/activemq/core/PrefetchPolicy.h> Inheritance diagram for activemq::core::PrefetchPolicy:

### Public Member Functions

- virtual **~PrefetchPolicy** ()
- virtual void **setDurableTopicPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int **getDurableTopicPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void **setQueuePrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Queue.*
- virtual int **getQueuePrefetch** () const =0  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void **setQueueBrowserPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int **getQueueBrowserPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void **setTopicPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Topic.*
- virtual int **getTopicPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int **getMaxPrefetchLimit** (int value) const =0  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual **PrefetchPolicy** \* **clone** () const =0  
*Clone the Policy and return a new pointer to that clone.*
- virtual void **configure** (const **decaf::util::Properties** &properties)  
*Checks the supplied properties object for properties matching the configurable settings of this class.*



## Protected Member Functions

- `PrefetchPolicy ()`

### 6.617.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

**Since:**

3.2.0

### 6.617.2 Constructor & Destructor Documentation

**6.617.2.1** `activemq::core::PrefetchPolicy::PrefetchPolicy ()` [protected]

**6.617.2.2** `virtual activemq::core::PrefetchPolicy::~~PrefetchPolicy ()` [virtual]

### 6.617.3 Member Function Documentation

**6.617.3.1** `virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone () const`  
[pure virtual]

Clone the Policy and return a new pointer to that clone.

**Returns:**

pointer to a new **PrefetchPolicy** (p. 2976) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1674).

**6.617.3.2** `virtual void activemq::core::PrefetchPolicy::configure (const decaf::util::Properties & properties)` [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class. The default implementation looks for properties named with the prefix `cms.PrefetchPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.PrefetchPolicy.topicPrefetch` will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

**Parameters:**

*properties* The Properties object used to configure this object.

**Exceptions:**

*NumberFormatException* if a property that is numeric cannot be converted

*IllegalArgumentException* if a property can't be converted to the correct type.

**6.617.3.3** `virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

**Returns:**

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1674).

**6.617.3.4** `virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit (int value) const [pure virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

**Returns:**

the allowable value for a prefetch limit, either requested or the max.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1674).

**6.617.3.5** `virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

**Returns:**

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1674).

**6.617.3.6** `virtual int activemq::core::PrefetchPolicy::getQueuePrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Queue.

**Returns:**

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1675).

**6.617.3.7** `virtual int activemq::core::PrefetchPolicy::getTopicPrefetch () const [pure virtual]`

Gets the amount of messages to prefetch for a Topic.

**Returns:**

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1675).

**6.617.3.8 virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch (int *value*)** [pure virtual]

Sets the amount of prefetched messages for a Durable Topic.

**Parameters:**

*value* The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1675).

**6.617.3.9 virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch (int *value*)** [pure virtual]

Sets the amount of prefetched messages for a Queue Browser.

**Parameters:**

*value* The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1675).

**6.617.3.10 virtual void activemq::core::PrefetchPolicy::setQueuePrefetch (int *value*)** [pure virtual]

Sets the amount of prefetched messages for a Queue.

**Parameters:**

*value* The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1676).

**6.617.3.11 virtual void activemq::core::PrefetchPolicy::setTopicPrefetch (int *value*)** [pure virtual]

Sets the amount of prefetched messages for a Topic.

**Parameters:**

*value* The number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1676).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**PrefetchPolicy.h**

## 6.618 activemq::util::PrimitiveList Class Reference

List of primitives.

#include <src/main/activemq/util/PrimitiveList.h> Inheritance diagram for activemq::util::PrimitiveList:

### Public Member Functions

- **PrimitiveList** ()  
*Default Constructor, creates an Empty list.*
- virtual ~**PrimitiveList** ()
- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)  
*Copy Constructor.*
- **PrimitiveList** (const **PrimitiveList** &src)  
*Copy Constructor.*
- std::string **toString** () const  
*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*
- virtual bool **getBool** (std::size\_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Boolean value at the specified index.*
- virtual void **setBool** (std::size\_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual unsigned char **getByte** (std::size\_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Byte value at the specified index.*
- virtual void **setByte** (std::size\_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual char **getChar** (std::size\_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Character value at the specified index.*

- virtual void **setChar** (std::size\_t index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual short **getShort** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Short value at the specified index.*
- virtual void **setShort** (std::size\_t index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual int **getInt** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Integer value at the specified index.*
- virtual void **setInt** (std::size\_t index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual long long **getLong** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Long value at the specified index.*
- virtual void **setLong** (std::size\_t index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual float **getFloat** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Float value at the specified index.*
- virtual void **setFloat** (std::size\_t index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual double **getDouble** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Double value at the specified index.*

- virtual void **setDouble** (std::size\_t index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual std::string **getString** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the String value at the specified index.*

- virtual void **setString** (std::size\_t index, const std::string &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual std::vector< unsigned char > **getByteArray** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Byte Array value at the specified index.*

- virtual void **setByteArray** (std::size\_t index, const std::vector< unsigned char > &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

## 6.618.1 Detailed Description

List of primitives.

## 6.618.2 Constructor & Destructor Documentation

### 6.618.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

### 6.618.2.2 virtual activemq::util::PrimitiveList::~~PrimitiveList () [virtual]

### 6.618.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & src)

Copy Constructor.

#### Parameters:

*src* - the Decaf List of PrimitiveNodeValues to copy

#### 6.618.2.4 activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & *src*)

Copy Constructor.

**Parameters:**

*src* - the **PrimitiveList** (p. 2980) to copy

### 6.618.3 Member Function Documentation

#### 6.618.3.1 virtual bool activemq::util::PrimitiveList::getBool (std::size\_t *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Boolean value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is > **size()** (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

#### 6.618.3.2 virtual unsigned char activemq::util::PrimitiveList::getBytes (std::size\_t *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Byte value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is > **size()** (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.3** `virtual std::vector<unsigned char> activemq::util::PrimitiveList::getByteArray (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Byte Array value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is > `size()` (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.4** `virtual char activemq::util::PrimitiveList::getChar (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Character value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is > `size()` (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.5** `virtual double activemq::util::PrimitiveList::getDouble (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Double value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index



**Exceptions:**

***IndexOutOfBoundsException*** if index is > **size()** (p. 3601)

***UnsupportedOperationException*** if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.6** virtual float activemq::util::PrimitiveList::getFloat (std::size\_t *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Float value at the specified index.

**Parameters:**

***index*** - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

***IndexOutOfBoundsException*** if index is > **size()** (p. 3601)

***UnsupportedOperationException*** if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.7** virtual int activemq::util::PrimitiveList::getInt (std::size\_t *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Integer value at the specified index.

**Parameters:**

***index*** - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

***IndexOutOfBoundsException*** if index is > **size()** (p. 3601)

***UnsupportedOperationException*** if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.8** virtual long long activemq::util::PrimitiveList::getLong (std::size\_t *index*)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Long value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is  $> \text{size}()$  (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.9** `virtual short activemq::util::PrimitiveList::getShort (std::size_t index)  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Short value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is  $> \text{size}()$  (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.10** `virtual std::string activemq::util::PrimitiveList::getString (std::size_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the String value at the specified index.

**Parameters:**

*index* - index to get value from

**Returns:**

value contained at the given index

**Exceptions:**

*IndexOutOfBoundsException* if index is  $> \text{size}()$  (p. 3601)

*UnsupportedOperationException* if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.11**    `virtual void activemq::util::PrimitiveList::setBool (std::size_t index,  
bool value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
)` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.12**    `virtual void activemq::util::PrimitiveList::setByte  
(std::size_t index, unsigned char value) throw (  
decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.13**    `virtual void activemq::util::PrimitiveList::setByteArray (std::size_t  
index, const std::vector< unsigned char > & value) throw (  
decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.14** `virtual void activemq::util::PrimitiveList::setChar (std::size_t index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.15** `virtual void activemq::util::PrimitiveList::setDouble (std::size_t index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.16** `virtual void activemq::util::PrimitiveList::setFloat (std::size_t index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.17** `virtual void activemq::util::PrimitiveList::setInt (std::size_t index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.18** `virtual void activemq::util::PrimitiveList::setLong (std::size_t index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.19** `virtual void activemq::util::PrimitiveList::setShort (std::size_t index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if index > size() (p. 3601).

**6.618.3.20** `virtual void activemq::util::PrimitiveList::setString  
(std::size_t index, const std::string & value) throw (  
decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters:**

*index* - location to set in the list

*value* - the new value to assign to the element at index

**Exceptions:**

*IndexOutOfBoundsException* if `index > size()` (p. 3601).

**6.618.3.21** `std::string activemq::util::PrimitiveList::toString ()` const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

**Returns:**

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveList.h`

## 6.619 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

#include <src/main/activemq/util/PrimitiveMap.h> Inheritance diagram for activemq::util::PrimitiveMap:

### Public Member Functions

- **PrimitiveMap** ()  
*Default Constructor, creates an empty map.*
- virtual ~**PrimitiveMap** ()
- **PrimitiveMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &source)  
*Copy Constructor.*
- **PrimitiveMap** (const **PrimitiveMap** &source)  
*Copy Constructor.*
- std::string **toString** () const  
*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*
- virtual bool **getBool** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setBool** (const std::string &key, bool value)  
*Sets the value at key to the specified type.*
- virtual unsigned char **getByte** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setByte** (const std::string &key, unsigned char value)  
*Sets the value at key to the specified type.*
- virtual char **getChar** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setChar** (const std::string &key, char value)

*Sets the value at key to the specified type.*

- virtual short **getShort** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, de-  
caf::lang::exceptions::UnsupportedOperationException )

*Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setShort** (const std::string &key, short value)

*Sets the value at key to the specified type.*

- virtual int **getInt** (const std::string &key) const throw ( de-  
caf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setInt** (const std::string &key, int value)

*Sets the value at key to the specified type.*

- virtual long long **getLong** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, de-  
caf::lang::exceptions::UnsupportedOperationException )

*Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setLong** (const std::string &key, long long value)

*Sets the value at key to the specified type.*

- virtual float **getFloat** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, de-  
caf::lang::exceptions::UnsupportedOperationException )

*Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setFloat** (const std::string &key, float value)

*Sets the value at key to the specified type.*

- virtual double **getDouble** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, de-  
caf::lang::exceptions::UnsupportedOperationException )

*Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setDouble** (const std::string &key, double value)

*Sets the value at key to the specified type.*

- virtual std::string **getString** (const std::string &key) const  
throw ( decaf::lang::exceptions::NoSuchElementException, de-  
caf::lang::exceptions::UnsupportedOperationException )

*Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*



- virtual void **setString** (const std::string &key, const std::string &value)

*Sets the value at key to the specified type.*

- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.*

- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)

*Sets the value at key to the specified type.*

## 6.619.1 Detailed Description

Map of named primitives.

## 6.619.2 Constructor & Destructor Documentation

### 6.619.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

### 6.619.2.2 virtual activemq::util::PrimitiveMap::~~PrimitiveMap () [virtual]

### 6.619.2.3 activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > & source)

Copy Constructor.

#### Parameters:

**source** The Decaf Library Map instance whose elements will be copied into this Map.

### 6.619.2.4 activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & source)

Copy Constructor.

#### Parameters:

**source** The **PrimitiveMap** (p. 2991) whose elements will be copied into this Map.

### 6.619.3 Member Function Documentation

**6.619.3.1** `virtual bool activemq::util::PrimitiveMap::getBool (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.2** `virtual unsigned char activemq::util::PrimitiveMap::getBytes (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.3** `virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getBytesArray (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at *key* in the type requested.

**Exceptions:**

*NoSuchElementException* if *key* is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.4** `virtual char activemq::util::PrimitiveMap::getChar (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Character value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at *key* in the type requested.

**Exceptions:**

*NoSuchElementException* if *key* is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.5** `virtual double activemq::util::PrimitiveMap::getDouble (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Double value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at *key* in the type requested.

**Exceptions:**

*NoSuchElementException* if *key* is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.6** `virtual float activemq::util::PrimitiveMap::getFloat (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.7** `virtual int activemq::util::PrimitiveMap::getInt (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.8** `virtual long long activemq::util::PrimitiveMap::getLong (const std::string & key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.9** virtual short activemq::util::PrimitiveMap::getShort (const std::string & *key*) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.10** virtual std::string activemq::util::PrimitiveMap::getString (const std::string & *key*) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

**Parameters:**

*key* - the location to return the value from.

**Returns:**

the value at key in the type requested.

**Exceptions:**

*NoSuchElementException* if key is not in the map.

*UnsupportedOperationException* if the value cannot be converted to the type this method returns

**6.619.3.11    virtual void activemq::util::PrimitiveMap::setBool (const std::string & *key*, bool *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.12    virtual void activemq::util::PrimitiveMap::setByte (const std::string & *key*, unsigned char *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.13    virtual void activemq::util::PrimitiveMap::setByteArray (const std::string & *key*, const std::vector< unsigned char > & *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.14    virtual void activemq::util::PrimitiveMap::setChar (const std::string & *key*, char *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.15    virtual void activemq::util::PrimitiveMap::setDouble (const std::string & *key*, double *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.16 virtual void activemq::util::PrimitiveMap::setFloat (const std::string & *key*, float *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.17 virtual void activemq::util::PrimitiveMap::setInt (const std::string & *key*, int *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.18 virtual void activemq::util::PrimitiveMap::setLong (const std::string & *key*, long long *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.19 virtual void activemq::util::PrimitiveMap::setShort (const std::string & *key*, short *value*) [virtual]**

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.20**    `virtual void activemq::util::PrimitiveMap::setString (const std::string & key, const std::string & value) [virtual]`

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

**Parameters:**

*key* - the map key to set or insert.

*value* - the new value to set at the key location.

**6.619.3.21**    `std::string activemq::util::PrimitiveMap::toString () const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

**Returns:**

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveMap.h`



## 6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference

This class wraps the functionality needed to **marshal** (p.107) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

### Public Member Functions

- **PrimitiveTypesMarshaller** ()
- virtual **~PrimitiveTypesMarshaller** ()

### Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** \*map, std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Marshal a primitive map object to the given byte buffer.*
- static void **unmarshal** (**util::PrimitiveMap** \*map, const std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveMap from the provided Byte buffer.*
- static void **marshal** (const **util::PrimitiveList** \*list, std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Marshal a primitive list object to the given byte buffer.*
- static void **unmarshal** (**util::PrimitiveList** \*list, const std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveList from the provided byte buffer.*
- static void **marshalMap** (const **util::PrimitiveMap** \*map, **decaf::io::DataOutputStream** &dataOut) throw ( decaf::lang::Exception )  
*Marshal a primitive map object to the given DataOutputStream.*
- static **util::PrimitiveMap** \* **unmarshalMap** (**decaf::io::DataInputStream** &dataIn) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveMap from the provided DataInputStream.*
- static void **marshalList** (const **util::PrimitiveList** \*list, **decaf::io::DataOutputStream** &dataOut) throw ( decaf::lang::Exception )  
*Marshal a PrimitiveList to the given DataOutputStream.*
- static **util::PrimitiveList** \* **unmarshalList** (**decaf::io::DataInputStream** &dataIn) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveList from the given DataInputStream.*

## Static Protected Member Functions

- static void **marshalPrimitiveMap** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::Map**< std::string, **util::PrimitiveValueNode** > &map) throw ( **decaf::io::IOException** )

*Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **marshalPrimitiveList** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::List**< **util::PrimitiveValueNode** > &list) throw ( **decaf::io::IOException** )

*Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static void **marshalPrimitive** (**decaf::io::DataOutputStream** &dataOut, const **util::PrimitiveValueNode** &value) throw ( **decaf::io::IOException** )

*Used to Marshal the Primitive types out on the Wire.*

- static void **unmarshalPrimitiveMap** (**decaf::io::DataInputStream** &dataIn, **util::PrimitiveMap** &map) throw ( **decaf::io::IOException** )

*Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **unmarshalPrimitiveList** (**decaf::io::DataInputStream** &dataIn, **decaf::util::StlList**< **util::PrimitiveValueNode** > &list) throw ( **decaf::io::IOException** )

*Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static **util::PrimitiveValueNode** **unmarshalPrimitive** (**decaf::io::DataInputStream** &dataIn) throw ( **decaf::io::IOException** )

*Unmarshals a Primitive Type from the stream, and returns it as a value Node.*

### 6.620.1 Detailed Description

This class wraps the functionality needed to **marshal** (p.107) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

## 6.620.2 Constructor & Destructor Documentation

6.620.2.1 `activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller()` [inline]

6.620.2.2 `virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller()` [inline, virtual]

## 6.620.3 Member Function Documentation

6.620.3.1 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal(const util::PrimitiveList * list, std::vector< unsigned char > & buffer) throw ( decaf::lang::Exception )` [static]

Marshal a primitive list object to the given byte buffer.

### Parameters:

*map* The PrimitiveList to Marshal.

*buffer* The byte buffer to write the marshaled data to.

### Exceptions:

*Exception* if an error occurs during the marshaling process.

6.620.3.2 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal(const util::PrimitiveMap * map, std::vector< unsigned char > & buffer) throw ( decaf::lang::Exception )` [static]

Marshal a primitive map object to the given byte buffer.

### Parameters:

*map* Map to Marshal.

*buffer* The byte buffer to write the marshaled data to.

### Exceptions:

*Exception* if an error occurs during the marshaling process.

6.620.3.3 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList(const util::PrimitiveList * list, decaf::io::DataOutputStream & dataOut) throw ( decaf::lang::Exception )` [static]

Marshal a PrimitiveList to the given DataOutputStream.

### Parameters:

*list* The list object to Marshal

*dataOut* Reference to a DataOutputStream to write the marshaled data to.

**Exceptions:**

*Exception* if an error occurs during the marshaling process.

```
6.620.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap
(const util::PrimitiveMap * map, decaf::io::DataOutputStream &
dataOut) throw ( decaf::lang::Exception ) [static]
```

Marshal a primitive map object to the given DataOutputStream.

**Parameters:**

*map* Map to Marshal.

*dataOut* Reference to a DataOutputStream to write the marshaled data to.

**Exceptions:**

*Exception* if an error occurs during the marshaling process.

```
6.620.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive
(decaf::io::DataOutputStream & dataOut, const util::PrimitiveValueNode
& value) throw ( decaf::io::IOException ) [static, protected]
```

Used to Marshal the Primitive types out on the Wire.

**Parameters:**

*dataOut* - the DataOutputStream to write to

*value* - the ValueNode to write.

**Exceptions:**

*IOException*

```
6.620.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList
(decaf::io::DataOutputStream & dataOut, const decaf::util::List<
util::PrimitiveValueNode > & list) throw ( decaf::io::IOException )
[static, protected]
```

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

**Parameters:**

*dataOut* - the DataOutputStream to write to

*list* - the ValueNode to write.

**Exceptions:**

*IOException*

**6.620.3.7** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap (decaf::io::DataOutputStream & *dataOut*, const decaf::util::Map< std::string, util::PrimitiveValueNode > & *map*) throw ( decaf::io::IOException ) [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

**Parameters:**

*dataOut* - the DataOutputStream to write to

*map* - the ValueNode to write.

**Exceptions:**

*IOException*

**6.620.3.8** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveList \* *list*, const std::vector< unsigned char > & *buffer*) throw ( decaf::lang::Exception ) [static]

Unmarshal a PrimitiveList from the provided byte buffer.

**Parameters:**

*map* The List to populate with values from the marshaled data.

*buffer* The byte buffer containing the marshaled Map.

**Exceptions:**

*Exception* if an error occurs during the unmarshal process.

**6.620.3.9** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveMap \* *map*, const std::vector< unsigned char > & *buffer*) throw ( decaf::lang::Exception ) [static]

Unmarshal a PrimitiveMap from the provided Byte buffer.

**Parameters:**

*map* The Map to populate with values from the marshaled data.

*buffer* The byte buffer containing the marshaled Map.

**Exceptions:**

*Exception* if an error occurs during the unmarshal process.

**6.620.3.10** `static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList (decaf::io::DataInputStream & dataIn) throw ( decaf::lang::Exception )`  
[static]

Unmarshal a PrimitiveList from the given DataInputStream.

**Parameters:**

*dataIn* The DataInputStream instance to read the marshaled PrimitiveList from.

**Returns:**

a pointer to a newly allocated PrimitiveList instnace.

**Exceptions:**

*Exception* if an error occurs during the unmarshal process.

**6.620.3.11** `static util::PrimitiveMap* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap (decaf::io::DataInputStream & dataIn) throw ( decaf::lang::Exception )`  
[static]

Unmarshal a PrimitiveMap from the provided DataInputStream.

**Parameters:**

*dataIn* The DataInputStream instance to read the marshaled PrimitiveMap from.

**Returns:**

a pointer to a newly allocated PrimitiveMap instnace.

**Exceptions:**

*Exception* if an error occurs during the unmarshal process.

**6.620.3.12** `static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & dataIn) throw ( decaf::io::IOException )`  
[static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

**Parameters:**

*dataIn* - DataInputStream to read from.

**Returns:**

a PrimitiveValueNode containing the data.

**Exceptions:**

*IOException*

**6.620.3.13** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & *dataIn*, decaf::util::StlList< util::PrimitiveValueNode > & *list*) throw ( decaf::io::IOException ) [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

**Parameters:**

*dataIn* - DataInputStream to read from.

*list* - the ValueNode to write.

**Exceptions:**

*IOException*

**6.620.3.14** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & *dataIn*, util::PrimitiveMap & *map*) throw ( decaf::io::IOException ) [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

**Parameters:**

*dataIn* - DataInputStream to read from.

*map* - the map to fill with data.

**Exceptions:**

*IOException*

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

## 6.621 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

### Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string \* **stringValue**
- std::vector< unsigned char > \* **byteArrayValue**
- decaf::util::List< PrimitiveValueNode > \* **listValue**
- decaf::util::Map< std::string, PrimitiveValueNode > \* **mapValue**

### 6.621.1 Detailed Description

Define a union type comprised of the various types.



## 6.621.2 Field Documentation

- 6.621.2.1 `bool activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue`
- 6.621.2.2 `std::vector<unsigned char>* activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue`
- 6.621.2.3 `unsigned char activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue`
- 6.621.2.4 `char activemq::util::PrimitiveValueNode::PrimitiveValue::charValue`
- 6.621.2.5 `double activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue`
- 6.621.2.6 `float activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue`
- 6.621.2.7 `int activemq::util::PrimitiveValueNode::PrimitiveValue::intValue`
- 6.621.2.8 `decaf::util::List<PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::listValue`
- 6.621.2.9 `long long activemq::util::PrimitiveValueNode::PrimitiveValue::longValue`
- 6.621.2.10 `decaf::util::Map<std::string, PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue`
- 6.621.2.11 `short activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue`
- 6.621.2.12 `std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue`

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

## 6.622 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p.3012) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

### Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- template<typename TO >  
TO **convert** (const **PrimitiveValueNode** &value) const throw ( decaf::lang::exceptions::UnsupportedOperationException )

### 6.622.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p.3012) from one type to another. If the conversion is supported then calling the convert method will throw an UnsupportedOperationException to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X X							
byte		X X X X X						
short			X X X X					
int				X X X X				
long					X X			
float						X X X		
double							X X X	
String								X X X X X X X

Since:

3.0

### 6.622.2 Constructor & Destructor Documentation

**6.622.2.1** **activemq::util::PrimitiveValueConverter::PrimitiveValueConverter** ()  
[inline]

**6.622.2.2** **virtual**  
**activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter** ()  
[inline, virtual]

### 6.622.3 Member Function Documentation

**6.622.3.1** **std::vector< unsigned char > ac-**  
**tivemq::util::PrimitiveValueConverter::convert** (const  
**PrimitiveValueNode** & *value*) const throw ( de-  
caf::lang::exceptions::UnsupportedOperationException )  
[inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

## 6.623 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

### Data Structures

- union **PrimitiveValue**

*Define a union type comprised of the various types.*

### Public Types

- enum **PrimitiveType** {  
    **NULL\_TYPE** = 0, **BOOLEAN\_TYPE** = 1, **BYTE\_TYPE** = 2, **CHAR\_TYPE** = 3,  
    **SHORT\_TYPE** = 4, **INTEGER\_TYPE** = 5, **LONG\_TYPE** = 6, **DOUBLE\_TYPE** = 7,  
    **FLOAT\_TYPE** = 8, **STRING\_TYPE** = 9, **BYTE\_ARRAY\_TYPE** = 10, **MAP\_TYPE** = 11,  
    **LIST\_TYPE** = 12, **BIG\_STRING\_TYPE** = 13 }

*Enumeration for the various primitive types.*

### Public Member Functions

- **PrimitiveValueNode** ()  
*Default Constructor, creates a value of the NULL\_TYPE.*
- **PrimitiveValueNode** (bool value)  
*Boolean Value Constructor.*
- **PrimitiveValueNode** (unsigned char value)  
*Byte Value Constructor.*
- **PrimitiveValueNode** (char value)  
*Char Value Constructor.*
- **PrimitiveValueNode** (short value)  
*Short Value Constructor.*
- **PrimitiveValueNode** (int value)  
*Int Value Constructor.*
- **PrimitiveValueNode** (long long value)  
*Long Value Constructor.*
- **PrimitiveValueNode** (float value)

*Float Value Constructor.*

- **PrimitiveValueNode** (double value)  
*Double Value Constructor.*
- **PrimitiveValueNode** (const char \*value)  
*String Value Constructor.*
- **PrimitiveValueNode** (const std::string &value)  
*String Value Constructor.*
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)  
*Byte Array Value Constructor.*
- **PrimitiveValueNode** (const decaf::util::List< PrimitiveValueNode > &value)  
*Primitive List Constructor.*
- **PrimitiveValueNode** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)  
*Primitive Map Value Constructor.*
- **PrimitiveValueNode** (const PrimitiveValueNode &node)  
*Copy constructor.*
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const PrimitiveValueNode &node)  
*Assignment operator, copies the data from the other node.*
- **bool operator==** (const PrimitiveValueNode &node) const  
*Comparison Operator, compares this node to the other node.*
- **PrimitiveType getType** () const  
*Gets the Value Type of this type wrapper.*
- **PrimitiveValue getValue** () const  
*Gets the internal Primitive Value object from this wrapper.*
- **void setValue** (const PrimitiveValue &value, PrimitiveType valueType)  
*Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.*
- **void clear** ()  
*Clears the value from this wrapper converting it back to a blank NULL\_ TYPE value.*
- **void setBool** (bool value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- **bool getBool** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Boolean value of this Node.*

- void **setByte** (unsigned char value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- unsigned char **getByte** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Byte value of this Node.*
- void **setChar** (char value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- char **getChar** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Character value of this Node.*
- void **setShort** (short value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- short **getShort** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Short value of this Node.*
- void **setInt** (int value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- int **getInt** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Integer value of this Node.*
- void **setLong** (long long value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- long long **getLong** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Long value of this Node.*
- void **setFloat** (float value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- float **getFloat** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Float value of this Node.*
- void **setDouble** (double value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- double **getDouble** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Double value of this Node.*

- void **setString** (const std::string &value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- std::string **getString** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the String value of this Node.*
- void **setByteArray** (const std::vector< unsigned char > &value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- std::vector< unsigned char > **getByteArray** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Byte Array value of this Node.*
- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- const decaf::util::List< PrimitiveValueNode > & **getList** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Primitive List value of this Node.*
- void **setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- const decaf::util::Map< std::string, PrimitiveValueNode > & **getMap** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Primitive Map value of this Node.*
- std::string **toString** () const  
*Creates a string representation of this value.*

### 6.623.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

### 6.623.2 Member Enumeration Documentation

#### 6.623.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

**NULL\_** **TYPE**

*BOOLEAN\_TYPE*  
*BYTE\_TYPE*  
*CHAR\_TYPE*  
*SHORT\_TYPE*  
*INTEGER\_TYPE*  
*LONG\_TYPE*  
*DOUBLE\_TYPE*  
*FLOAT\_TYPE*  
*STRING\_TYPE*  
*BYTE\_ARRAY\_TYPE*  
*MAP\_TYPE*  
*LIST\_TYPE*  
*BIG\_STRING\_TYPE*

### 6.623.3 Constructor & Destructor Documentation

#### 6.623.3.1 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ()`

Default Constructor, creates a value of the `NULL_TYPE`.

#### 6.623.3.2 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool value)`

Boolean Value Constructor.

**Parameters:**

*value* - the new value to store.

#### 6.623.3.3 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char value)`

Byte Value Constructor.

**Parameters:**

*value* - the new value to store.

#### 6.623.3.4 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (char value)`

Char Value Constructor.

**Parameters:**

*value* - the new value to store.



**6.623.3.5** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (short value)`

Short Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.6** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (int value)`

Int Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.7** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long value)`

Long Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.8** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (float value)`

Float Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.9** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (double value)`

Double Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.10** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * value)`

String Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.11** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & value)`

String Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.12** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & value)`

Byte Array Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.13** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & value)`

Primitive List Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.14** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Primitive Map Value Constructor.

**Parameters:**

*value* - the new value to store.

**6.623.3.15** `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & node)`

Copy constructor.

**Parameters:**

*node* The instance of another node to copy to this one.

**6.623.3.16** `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode () [inline]`

## 6.623.4 Member Function Documentation

**6.623.4.1** `void activemq::util::PrimitiveValueNode::clear ()`

Clears the value from this wrapper converting it back to a blank NULL\_TYPE value.

#### 6.623.4.2 `bool activemq::util::PrimitiveValueNode::getBool () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Boolean value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

#### 6.623.4.3 `unsigned char activemq::util::PrimitiveValueNode::getBytes () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

#### 6.623.4.4 `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getBytes () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte Array value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

#### 6.623.4.5 `char activemq::util::PrimitiveValueNode::getChar () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Character value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.6    double activemq::util::PrimitiveValueNode::getDouble () const throw (decaf::lang::exceptions::NoSuchElementException )**

Gets the Double value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.7    float activemq::util::PrimitiveValueNode::getFloat () const throw ( decaf::lang::exceptions::NoSuchElementException )**

Gets the Float value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.8    int activemq::util::PrimitiveValueNode::getInt () const throw ( decaf::lang::exceptions::NoSuchElementException )**

Gets the Integer value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.9    const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList () const throw ( decaf::lang::exceptions::NoSuchElementException )**

Gets the Primitive List value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.10** `long long activemq::util::PrimitiveValueNode::getLong () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Long value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.11** `const decaf::util::Map<std::string, PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getMap () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Primitive Map value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.12** `short activemq::util::PrimitiveValueNode::getShort () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Short value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.13** `std::string activemq::util::PrimitiveValueNode::getString () const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the String value of this Node.

**Returns:**

value contained at the given index

**Exceptions:**

*NoSuchElementException* this node cannot be returned as the requested type.

**6.623.4.14**   **PrimitiveType** activemq::util::PrimitiveValueNode::getType () const  
[inline]

Gets the Value Type of this type wrapper.

**Returns:**

the PrimitiveType value for this wrapper.

**6.623.4.15**   **PrimitiveValue** activemq::util::PrimitiveValueNode::getValue () const  
[inline]

Gets the internal Primitive Value object from this wrapper.

**Returns:**

a copy of the contained **PrimitiveValue** (p. 3008)

**6.623.4.16**   **PrimitiveValueNode&** activemq::util::PrimitiveValueNode::operator=  
(const PrimitiveValueNode & *node*)

Assignment operator, copies the data from the other node.

**Parameters:**

*node* The instance of another node to copy to this one.

**6.623.4.17**   **bool** activemq::util::PrimitiveValueNode::operator== (const  
PrimitiveValueNode & *node*) const

Comparison Operator, compares this node to the other node.

**Returns:**

true if the values are the same false otherwise.

**6.623.4.18**   **void** activemq::util::PrimitiveValueNode::setBool (bool *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.19**   **void** activemq::util::PrimitiveValueNode::setByte (unsigned char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.20 void activemq::util::PrimitiveValueNode::setByteArray (const std::vector< unsigned char > & *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.21 void activemq::util::PrimitiveValueNode::setChar (char *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.22 void activemq::util::PrimitiveValueNode::setDouble (double *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.23 void activemq::util::PrimitiveValueNode::setFloat (float *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.24 void activemq::util::PrimitiveValueNode::setInt (int *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.25 void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.26 void activemq::util::PrimitiveValueNode::setLong (long long *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.27 void activemq::util::PrimitiveValueNode::setMap (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.28 void activemq::util::PrimitiveValueNode::setShort (short *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.29 void activemq::util::PrimitiveValueNode::setString (const std::string & *value*)**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters:**

*value* - the new value to assign to the element at index

**6.623.4.30 void activemq::util::PrimitiveValueNode::setValue (const PrimitiveValue & *value*, PrimitiveType *valueType*)**

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

**Parameters:**

*value* The value to set as the value contained in this Node.

*valueType* The type of the value being set into this one.



**6.623.4.31** `std::string activemq::util::PrimitiveValueNode::toString () const`

Creates a string representation of this value.

**Returns:**

string value of this type wrapper.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

## 6.624 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

```
#include <src/main/decaf/security/Principal.h>Inheritance    diagram    for    de-  
caf::security::Principal:
```

### Public Member Functions

- virtual **~Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0  
*Compares two principals to see if they are the same.*
- virtual std::string **getName** () const =0  
*Provides the name of this principal.*

### 6.624.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

### 6.624.2 Constructor & Destructor Documentation

**6.624.2.1** virtual decaf::security::Principal::~~Principal () [inline, virtual]

### 6.624.3 Member Function Documentation

**6.624.3.1** virtual bool decaf::security::Principal::equals (const **Principal** & *another*) const [pure virtual]

Compares two principals to see if they are the same.

#### Parameters:

*another* A principal to be tested for equality to this one.

#### Returns:

true if the given principal is equivalent to this one.

**6.624.3.2** virtual std::string decaf::security::Principal::getName () const [pure virtual]

Provides the name of this principal.

#### Returns:

the name of this principal.

Implemented in **decaf::security::auth::x500::X500Principal** (p. 4027).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Principal.h**

## 6.625 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

#include <src/main/decaf/util/PriorityQueue.h> Inheritance diagram for decaf::util::PriorityQueue< E >:

### Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

### Public Member Functions

- **PriorityQueue** ()  
*Creates a **Priority Queue** (p. 3149) with the default initial capacity.*
- **PriorityQueue** (std::size\_t initialCapacity)  
*Creates a **Priority Queue** (p. 3149) with the capacity value supplied.*
- **PriorityQueue** (std::size\_t initialCapacity, **Comparator**< E > \*comparator)  
*Creates a **Priority Queue** (p. 3149) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)  
*Creates a **PriorityQueue** (p. 3028) containing the elements in the specified **Collection** (p. 1184).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)  
*Creates a **PriorityQueue** (p. 3028) containing the elements in the specified priority queue.*
- virtual ~**PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (const **Collection**< E > &source)  
*Assignment operator, assign another **Collection** (p. 1184) to this one.*
- **PriorityQueue**< E > & **operator=** (const **PriorityQueue**< E > &source)  
*Assignment operator, assign another **PriorityQueue** (p. 3028) to this one.*
- virtual **decaf::util::Iterator**< E > \* **iterator** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

*Inserts the specified element into the queue provided that the condition allows such an operation.*

- virtual bool **poll** (E &result)  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const  
*Gets but not removes the element in the head of the queue.*
- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves and removes the head of this queue.*
- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from this collection, if it is present (optional operation).*
- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.*
- decaf::lang::Pointer< Comparator< E > > **comparator** () const  
*obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 3028) is using to compare the elements in the queue with.*

## Friends

- class **PriorityQueueIterator**

### 6.625.1 Detailed Description

**template<typename E> class decaf::util::PriorityQueue< E >**

An unbounded priority queue based on a binary heap algorithm. The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 1217) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an **internal** (p. 144) capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 1184) and **Iterator** (p. 2154) interfaces. The **Iterator** (p. 2154) provided in method **iterator()** (p. 3032) is

not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort( pq.toArray() )`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 3028) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides  $O(\log(n))$  time for the enqueueing and dequeuing methods (`offer`, `poll`, **`remove()`** (p. 3034) and `add`); linear time for the `remove(Object)` and `contains(Object)` methods; and constant time for the retrieval methods (`peek`, `element`, and `size`).

**Since:**

1.0

## 6.625.2 Constructor & Destructor Documentation

**6.625.2.1** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue  
( ) [inline]`

Creates a **Priority Queue** (p. 3149) with the default initial capacity.

**6.625.2.2** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue  
(std::size_t initialCapacity) [inline]`

Creates a **Priority Queue** (p. 3149) with the capacity value supplied.

**Parameters:**

*initialCapacity* The initial number of elements allocated to this **PriorityQueue** (p. 3028).

**6.625.2.3** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue  
(std::size_t initialCapacity, Comparator< E > * comparator) [inline]`

Creates a **Priority Queue** (p. 3149) with the default initial capacity. This new **PriorityQueue** (p. 3028) takes ownership of the passed **Comparator** (p. 1217) instance and uses that to determine the ordering of the elements in the **Queue** (p. 3149).

**Parameters:**

*initialCapacity* The initial number of elements allocated to this **PriorityQueue** (p. 3028).

*comparator* The **Comparator** (p. 1217) instance to use in sorting the elements in the **Queue** (p. 3149).

**Exceptions:**

*NullPointerException* if the passed **Comparator** (p. 1217) is NULL.

**6.625.2.4** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue  
(const Collection< E > & source) [inline]`

Creates a **PriorityQueue** (p. 3028) containing the elements in the specified **Collection** (p. 1184).

**Parameters:**

*source* the **Collection** (p. 1184) whose elements are to be placed into this priority queue

**6.625.2.5** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const PriorityQueue< E > & source) [inline]`

Creates a **PriorityQueue** (p. 3028) containing the elements in the specified priority queue. This priority queue will be ordered according to the same ordering as the given priority queue.

**Parameters:**

*source* the priority queue whose elements are to be placed into this priority queue

**6.625.2.6** `template<typename E> virtual decaf::util::PriorityQueue< E >::~~PriorityQueue () [inline, virtual]`

**6.625.3 Member Function Documentation**

**6.625.3.1** `template<typename E> virtual bool decaf::util::PriorityQueue< E >::add (const E & value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available. This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

**Parameters:**

*value* - the element to offer to the **Queue** (p. 3149).

**Returns:**

true if the add succeeds.

**Exceptions:**

*IllegalArgumentException* if the element cannot be added.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 195).

References `DECAF_CATCH_EXCEPTION_CONVERT`, `DECAF_CATCH_RETHROW`, `DECAF_CATCHALL_THROW`, and `decaf::util::PriorityQueue< E >::offer()`.

**6.625.3.2** `template<typename E> virtual void decaf::util::PriorityQueue< E >::clear () throw ( lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all elements of the queue. This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 196).

**6.625.3.3** `template<typename E> decaf::lang::Pointer< Comparator<E> >  
decaf::util::PriorityQueue< E >::comparator () const [inline]`

obtains a Copy of the Pointer instance that this **PriorityQueue** (p.3028) is using to compare the elements in the queue with. The returned value is a copy, the caller cannot change the value if the **internal** (p.144) Pointer value.

**Returns:**

a copy of the **Comparator** (p.1217) Pointer being used by this **Queue** (p.3149).

**6.625.3.4** `template<typename E> virtual decaf::util::Iterator<E>*  
decaf::util::PriorityQueue< E >::iterator () const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.2152).

**6.625.3.5** `template<typename E> virtual decaf::util::Iterator<E>*  
decaf::util::PriorityQueue< E >::iterator () [inline, virtual]`

**Returns:**

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p.2152).

References **decaf::util::PriorityQueue< E >::PriorityQueueIterator**.

**6.625.3.6** `template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer  
(const E & value) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation. The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

**Parameters:**

*value* the specified element to insert into the queue.

**Returns:**

true if the operation succeeds and false if it fails.

**Exceptions:**

**NullPointerException** if the **Queue** (p.3149) implementation does not allow Null values to be inserted into the **Queue** (p.3149).

**IllegalArgumentException** if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p.3150).

Referenced by **decaf::util::PriorityQueue< E >::add()**.



**6.625.3.7** `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const PriorityQueue< E > & source) [inline]`

Assignment operator, assign another **PriorityQueue** (p. 3028) to this one.

**Parameters:**

*source* The **PriorityQueue** (p. 3028) to copy to this one.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 187).

**6.625.3.8** `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const Collection< E > & source) [inline]`

Assignment operator, assign another **Collection** (p. 1184) to this one.

**Parameters:**

*source* The **Collection** (p. 1184) to copy to this one.

**6.625.3.9** `template<typename E> virtual bool decaf::util::PriorityQueue< E>::peek (E & result) const [inline, virtual]`

Gets but not removes the element in the head of the queue. The result if successful is assigned to the result parameter.

**Parameters:**

*result* Reference to an instance of the contained type to assigned the removed value to.

**Returns:**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 3151).

**6.625.3.10** `template<typename E> virtual bool decaf::util::PriorityQueue< E>::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue. If the operation succeeds the value of the element at the head of the **Queue** (p. 3149) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

**Parameters:**

*result* Reference to an instance of the contained type to assigned the removed value to.

**Returns:**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 3151).

**6.625.3.11** `template<typename E> virtual bool decaf::util::PriorityQueue<E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters:**

*value* - element to be removed from this collection, if present

**Returns:**

true if an element was removed as a result of this call

**Exceptions:**

*UnsupportedOperationException* if the remove operation is not supported by this collection.

*IllegalArgumentException* If the value is not a valid entry for this **Collection** (p.1184).

Reimplemented from `decaf::util::AbstractCollection< E >` (p.187).

**6.625.3.12** `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove () throw ( decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Retrieves and removes the head of this queue. This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

**Returns:**

a copy of the element in the head of the queue.

**Exceptions:**

*NoSuchElementException* if the queue is empty.

Reimplemented from `decaf::util::AbstractQueue< E >` (p.196).

**6.625.3.13** `template<typename E> virtual std::size_t decaf::util::PriorityQueue< E >::size () const [inline, virtual]`

Returns the number of elements in this collection. If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

**Returns:**

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 1192).

**6.625.4 Friends And Related Function Documentation****6.625.4.1 template<typename E> friend class PriorityQueueIterator [friend]**

Referenced by decaf::util::PriorityQueue< E >::iterator().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**PriorityQueue.h**

## 6.626 activemq::commands::ProducerAck Class Reference

`#include <src/main/activemq/commands/ProducerAck.h>` Inheritance diagram for `activemq::commands::ProducerAck`:

### Public Member Functions

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const
- virtual void **setSize** (int size)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_PRODUCERACK** = 19

### Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- int **size**

## 6.626.1 Constructor & Destructor Documentation

**6.626.1.1** `activemq::commands::ProducerAck::ProducerAck ()`

**6.626.1.2** `virtual activemq::commands::ProducerAck::~~ProducerAck ()` [virtual]

## 6.626.2 Member Function Documentation

**6.626.2.1** `virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.626.2.2** `virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.626.2.3** `virtual bool activemq::commands::ProducerAck::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.626.2.4** `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

- 6.626.2.5** `virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId ()`  
[virtual]
- 6.626.2.6** `virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const`  
[virtual]
- 6.626.2.7** `virtual int activemq::commands::ProducerAck::getSize () const`  
[virtual]
- 6.626.2.8** `virtual bool activemq::commands::ProducerAck::isProducerAck () const`  
[inline, virtual]

**Returns:**

an answer of true to the **isProducerAck()** (p. 3038) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 762).

- 6.626.2.9** `virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.626.2.10** `virtual void activemq::commands::ProducerAck::setSize (int size)`  
[virtual]
- 6.626.2.11** `virtual std::string activemq::commands::ProducerAck::toString () const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.626.2.12** `virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.626.3 Field Documentation

**6.626.3.1** `const unsigned char activemq::commands::ProducerAck::ID_PRODUCERACK = 19` [static]

**6.626.3.2** `Pointer<ProducerId> activemq::commands::ProducerAck::producerId` [protected]

**6.626.3.3** `int activemq::commands::ProducerAck::size` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

## 6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3040).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.627.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3040).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.627.2 Constructor & Destructor Documentation

**6.627.2.1** `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller()` `[inline]`

**6.627.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller()` `[inline, virtual]`

## 6.627.3 Member Function Documentation

**6.627.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.627.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.627.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.627.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.627.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.627.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.627.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**

## 6.628 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3044).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.628.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3044).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.628.2 Constructor & Destructor Documentation

**6.628.2.1** `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

**6.628.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

## 6.628.3 Member Function Documentation

**6.628.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.628.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.628.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.628.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.628.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.628.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.628.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerAckMarshaller.h**

## 6.629 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3048).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.629.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3048).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.629.2 Constructor & Destructor Documentation

**6.629.2.1** `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

**6.629.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

## 6.629.3 Member Function Documentation

**6.629.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.629.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.629.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.629.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.629.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.629.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.629.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerAckMarshaller.h**

## 6.630 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3052).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.630.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3052).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.630.2 Constructor & Destructor Documentation

**6.630.2.1** `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

**6.630.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

## 6.630.3 Member Function Documentation

**6.630.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.630.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.630.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.630.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.630.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.630.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.630.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerAckMarshaller.h**

## 6.631 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3056).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.631.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3056).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.631.2 Constructor & Destructor Documentation

**6.631.2.1** `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

**6.631.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

## 6.631.3 Member Function Documentation

**6.631.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.631.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.631.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.631.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.631.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.631.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.631.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerAckMarshaller.h**

## 6.632 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3060).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.632.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerAckMarshaller** (p.3060).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.632.2 Constructor & Destructor Documentation

**6.632.2.1** `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

**6.632.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

## 6.632.3 Member Function Documentation

**6.632.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.632.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.632.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.632.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.632.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.632.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.632.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**

## 6.633 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

#include <src/main/activemq/cmsutil/ProducerCallback.h> Inheritance diagram for activemq::cmsutil::ProducerCallback:

### Public Member Functions

- virtual `~ProducerCallback ()`
- virtual void `doInCms (cms::Session *session, cms::MessageProducer *producer)=0`  
throw ( cms::CMSException )

*Execute an action given a session and producer.*

### 6.633.1 Detailed Description

Callback for sending a message to a CMS destination.

### 6.633.2 Constructor & Destructor Documentation

- 6.633.2.1** virtual `activemq::cmsutil::ProducerCallback::~~ProducerCallback ()`  
[inline, virtual]

### 6.633.3 Member Function Documentation

- 6.633.3.1** virtual void `activemq::cmsutil::ProducerCallback::doInCms`  
(cms::Session \* *session*, cms::MessageProducer \* *producer*) throw ( cms::CMSException ) [pure virtual]

Execute an action given a session and producer.

#### Parameters:

*session* the CMS Session

*producer* the CMS Producer

#### Exceptions:

*cms::CMSException* (p. 1160) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::SendExecutor` (p. 3350).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/ProducerCallback.h



## 6.634 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ProducerExecutor:

### Public Member Functions

- **ProducerExecutor** (**ProducerCallback** \**action*, **CmsTemplate** \**parent*, **cms::Destination** \**destination*)
- virtual **~ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** \**session*) throw ( **cms::CMSEException** )  
*Execute any number of operations against the supplied CMS session.*
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \**session* **AMQCPP\_UNUSED**) throw ( **cms::CMSEException** )

### Protected Member Functions

- **ProducerExecutor** (const **ProducerExecutor** &)
- **ProducerExecutor** & **operator=** (const **ProducerExecutor** &)

### Protected Attributes

- **ProducerCallback** \* *action*
- **CmsTemplate** \* *parent*
- **cms::Destination** \* *destination*

#### 6.634.1 Constructor & Destructor Documentation

- 6.634.1.1 **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (const **ProducerExecutor** &) [inline, protected]
- 6.634.1.2 **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (**ProducerCallback** \* *action*, **CmsTemplate** \* *parent*, **cms::Destination** \* *destination*) [inline]
- 6.634.1.3 virtual **activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor** () [inline, virtual]

#### 6.634.2 Member Function Documentation

- 6.634.2.1 virtual void **activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms** (**cms::Session** \* *session*) throw ( **cms::CMSEException** ) [virtual]

Execute any number of operations against the supplied CMS session.

**Parameters:**

*session* the CMS Session

**Exceptions:**

*cms::CMSException* (p. 1160) if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 3378).

**6.634.2.2** `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session AMQCPP_UNUSED) throw ( cms::CMSException )` [inline, virtual]

**6.634.2.3** `ProducerExecutor& activemq::cmsutil::CmsTemplate::ProducerExecutor::operator= (const ProducerExecutor &)` [inline, protected]

Reimplemented in **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor** (p. 3278).

**6.634.3 Field Documentation**

**6.634.3.1** `ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action` [protected]

**6.634.3.2** `cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination` [protected]

**6.634.3.3** `CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.635 activemq::commands::ProducerId Class Reference

#include <src/main/activemq/commands/ProducerId.h> Inheritance diagram for activemq::commands::ProducerId:

### Public Types

- typedef decaf::lang::PointerComparator< ProducerId > COMPARATOR

### Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- **ProducerId** (std::string producerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- void **setProducerSessionKey** (std::string sessionKey)
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual int **compareTo** (const **ProducerId** &value) const
- virtual bool **equals** (const **ProducerId** &value) const
- virtual bool **operator==** (const **ProducerId** &value) const
- virtual bool **operator<** (const **ProducerId** &value) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

## Static Public Attributes

- static const unsigned char **ID\_PRODUCERID** = 123

## Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

### 6.635.1 Member Typedef Documentation

- 6.635.1.1** `typedef decaf::lang::PointerComparator<ProducerId>  
activemq::commands::ProducerId::COMPARATOR`

### 6.635.2 Constructor & Destructor Documentation

- 6.635.2.1** `activemq::commands::ProducerId::ProducerId ()`
- 6.635.2.2** `activemq::commands::ProducerId::ProducerId (const ProducerId &  
other)`
- 6.635.2.3** `activemq::commands::ProducerId::ProducerId (const SessionId &  
sessionId, long long consumerId)`
- 6.635.2.4** `activemq::commands::ProducerId::ProducerId (std::string producerId)`
- 6.635.2.5** `virtual activemq::commands::ProducerId::~~ProducerId () [virtual]`

### 6.635.3 Member Function Documentation

- 6.635.3.1** `virtual ProducerId* ac-  
tivemq::commands::ProducerId::cloneDataStructure ()  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p.1660).

- 6.635.3.2** `virtual int activemq::commands::ProducerId::compareTo (const  
ProducerId & value) const [virtual]`
- 6.635.3.3** `virtual void activemq::commands::ProducerId::copyDataStructure (const  
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

**6.635.3.4** virtual bool activemq::commands::ProducerId::equals (const ProducerId & *value*) const [virtual]

**6.635.3.5** virtual bool activemq::commands::ProducerId::equals (const DataStructure \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

**6.635.3.6** virtual std::string& activemq::commands::ProducerId::getConnectionId () [virtual]

**6.635.3.7** virtual const std::string& activemq::commands::ProducerId::getConnectionId () const [virtual]

**6.635.3.8** virtual unsigned char activemq::commands::ProducerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.635.3.9 `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const`
- 6.635.3.10 `virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]`
- 6.635.3.11 `virtual long long activemq::commands::ProducerId::getValue () const [virtual]`
- 6.635.3.12 `virtual bool activemq::commands::ProducerId::operator< (const ProducerId & value) const [virtual]`
- 6.635.3.13 `ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & other)`
- 6.635.3.14 `virtual bool activemq::commands::ProducerId::operator== (const ProducerId & value) const [virtual]`
- 6.635.3.15 `virtual void activemq::commands::ProducerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.635.3.16 `void activemq::commands::ProducerId::setProducerSessionKey (std::string sessionKey)`
- 6.635.3.17 `virtual void activemq::commands::ProducerId::setSessionId (long long sessionId) [virtual]`
- 6.635.3.18 `virtual void activemq::commands::ProducerId::setValue (long long value) [virtual]`
- 6.635.3.19 `virtual std::string activemq::commands::ProducerId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.833).

### 6.635.4 Field Documentation

- 6.635.4.1 `std::string activemq::commands::ProducerId::connectionId [protected]`
- 6.635.4.2 `const unsigned char activemq::commands::ProducerId::ID__ - PRODUCERID = 123 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

**6.635.4.3**   `long long activemq::commands::ProducerId::sessionId`   [protected]

**6.635.4.4**   `long long activemq::commands::ProducerId::value`   [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerId.h`

## 6.636 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3072).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.636.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3072). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.636.2 Constructor & Destructor Documentation

**6.636.2.1** `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.636.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.636.3 Member Function Documentation

**6.636.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.636.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.636.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.636.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.636.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.636.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.636.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h**

## 6.637 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3076).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.637.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3076). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.637.2 Constructor & Destructor Documentation

**6.637.2.1** `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.637.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.637.3 Member Function Documentation

**6.637.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.637.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.637.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.637.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.637.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.637.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.637.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerIdMarshaller.h**

## 6.638 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3080).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.638.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3080). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.638.2 Constructor & Destructor Documentation

**6.638.2.1** `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.638.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.638.3 Member Function Documentation

**6.638.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.638.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.638.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.638.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.638.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.638.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.638.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerIdMarshaller.h**

## 6.639 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3084).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.639.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3084). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.639.2 Constructor & Destructor Documentation

**6.639.2.1** `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.639.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.639.3 Member Function Documentation

**6.639.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.639.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.639.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.639.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.639.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.639.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.639.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerIdMarshaller.h**

## 6.640 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3088).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.640.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3088). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.640.2 Constructor & Destructor Documentation

**6.640.2.1** `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.640.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.640.3 Member Function Documentation

**6.640.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.640.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.640.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.640.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.640.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.640.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.640.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerIdMarshaller.h**

## 6.641 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3092).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.641.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3092). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.641.2 Constructor & Destructor Documentation

**6.641.2.1** `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

**6.641.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

## 6.641.3 Member Function Documentation

**6.641.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.641.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.641.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.641.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.641.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.641.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.641.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h**

## 6.642 activemq::commands::ProducerInfo Class Reference

#include <src/main/activemq/commands/ProducerInfo.h> Inheritance diagram for activemq::commands::ProducerInfo:

### Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ProducerId > & getProducerId** () const
- virtual **Pointer< ProducerId > & getProducerId** ()
- virtual void **setProducerId** (const **Pointer< ProducerId > &producerId**)
- virtual const **Pointer< ActiveMQDestination > & getDestination** () const
- virtual **Pointer< ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int windowSize)
- virtual bool **isProducerInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*



## Static Public Attributes

- static const unsigned char **ID\_PRODUCERINFO** = 6

## Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- **std::vector< decaf::lang::Pointer< BrokerId > > brokerPath**
- **bool dispatchAsync**
- **int windowSize**

## 6.642.1 Constructor & Destructor Documentation

**6.642.1.1** **activemq::commands::ProducerInfo::ProducerInfo ()**

**6.642.1.2** **virtual activemq::commands::ProducerInfo::~~ProducerInfo ()** [virtual]

## 6.642.2 Member Function Documentation

**6.642.2.1** **virtual ProducerInfo\* activemq::commands::ProducerInfo::cloneDataStructure ()**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1660).

**6.642.2.2** **virtual void activemq::commands::ProducerInfo::copyDataStructure (const DataStructure \* src)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 759).

**6.642.2.3** **Pointer<RemoveInfo> activemq::commands::ProducerInfo::createRemoveCommand ()** **const**

**6.642.2.4** **virtual bool activemq::commands::ProducerInfo::equals (const DataStructure \* value)** **const** [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are

the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.642.2.5** `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]`

**6.642.2.6** `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]`

**6.642.2.7** `virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

**6.642.2.8** `virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () [virtual]`

**6.642.2.9** `virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () const [virtual]`

**6.642.2.10** `virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () [virtual]`

**6.642.2.11** `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () const [virtual]`

**6.642.2.12** `virtual int activemq::commands::ProducerInfo::getWindowSize () const [virtual]`

**6.642.2.13** `virtual bool activemq::commands::ProducerInfo::isDispatchAsync () const [virtual]`

**6.642.2.14** `virtual bool activemq::commands::ProducerInfo::isProducerInfo () const [inline, virtual]`

**Returns:**

an answer of true to the `isProducerInfo()` (p. 3098) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 762).

- 6.642.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)` [virtual]
- 6.642.2.16 `virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.642.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool dispatchAsync)` [virtual]
- 6.642.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.642.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize (int windowSize)` [virtual]
- 6.642.2.20 `virtual std::string activemq::commands::ProducerInfo::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.642.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

### 6.642.3 Field Documentation

- 6.642.3.1 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::ProducerInfo::brokerPath` [protected]
- 6.642.3.2 `Pointer<ActiveMQDestination>` `activemq::commands::ProducerInfo::destination`  
[protected]
- 6.642.3.3 `bool` `activemq::commands::ProducerInfo::dispatchAsync` [protected]
- 6.642.3.4 `const unsigned char` `activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6` [static]
- 6.642.3.5 `Pointer<ProducerId>` `activemq::commands::ProducerInfo::producerId`  
[protected]
- 6.642.3.6 `int` `activemq::commands::ProducerInfo::windowSize` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerInfo.h`

## 6.643 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3101).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.643.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3101).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.643.2 Constructor & Destructor Documentation

**6.643.2.1** `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.643.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.643.3 Member Function Documentation

**6.643.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.643.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.643.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.643.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.643.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.643.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.643.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h`



## 6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3105).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.644.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3105).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.644.2 Constructor & Destructor Documentation

**6.644.2.1** `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.644.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.644.3 Member Function Documentation

**6.644.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.644.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.644.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 801).

**6.644.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 802).

**6.644.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 803).

**6.644.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.644.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**

## 6.645 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3109).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.645.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3109).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.645.2 Constructor & Destructor Documentation

**6.645.2.1** `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.645.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.645.3 Member Function Documentation

**6.645.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.645.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.645.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.645.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.645.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.645.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.645.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**



## 6.646 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3113).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.646.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3113).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.646.2 Constructor & Destructor Documentation

**6.646.2.1** `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.646.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.646.3 Member Function Documentation

**6.646.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.646.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.646.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 787).

**6.646.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 788).

**6.646.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 789).

**6.646.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.646.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**

## 6.647 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3117).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.647.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3117).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.647.2 Constructor & Destructor Documentation

**6.647.2.1** `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.647.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.647.3 Member Function Documentation

**6.647.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.647.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.647.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.647.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.647.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.647.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.647.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**



## 6.648 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3121).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.648.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **ProducerInfoMarshaller** (p.3121).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.648.2 Constructor & Destructor Documentation

**6.648.2.1** `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

**6.648.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

## 6.648.3 Member Function Documentation

**6.648.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.648.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.648.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.648.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.648.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.648.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.648.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerInfoMarshaller.h**

## 6.649 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

### Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual **~ProducerState** ()
- **std::string toString** () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const
- void **setTransactionState** (const **Pointer**< **TransactionState** > &transactionState)
- **Pointer**< **TransactionState** > **getTransactionState** () const

### 6.649.1 Constructor & Destructor Documentation

**6.649.1.1** **activemq::state::ProducerState::ProducerState** (const **Pointer**< **ProducerInfo** > & *info*)

**6.649.1.2** virtual **activemq::state::ProducerState::~~ProducerState** () [virtual]

### 6.649.2 Member Function Documentation

**6.649.2.1** const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** () const [inline]

**6.649.2.2** **Pointer**<**TransactionState**> **activemq::state::ProducerState::getTransactionState** () const

**6.649.2.3** void **activemq::state::ProducerState::setTransactionState** (const **Pointer**< **TransactionState** > & *transactionState*)

**6.649.2.4** **std::string** **activemq::state::ProducerState::toString** () const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ProducerState.h`

## 6.650 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

### Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)  
*Assignment Operator.*
- bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- std::size\_t **size** () const
- const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- std::string **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- std::string **remove** (const std::string &name)  
*Removes the property with the given name.*
- std::vector< std::string > **propertyNames** () const  
*Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.*
- std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- void **copy** (const **Properties** &source)  
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 3126) instance in NULL then this **List** (p. 2337) is not modified.*
- **Properties** \* **clone** () const  
*Clones this object.*
- void **clear** ()  
*Clears all properties from the map.*

- **bool equals** (const **Properties** &source) const  
*Test whether two **Properties** (p. 3126) objects are equivalent.*
- **std::string toString** () const  
*Formats the contents of the **Properties** (p. 3126) Object into a string that can be logged, etc.*
- **void load** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Reads a property list (key and element pairs) from the input byte stream.*
- **void load** (decaf::io::Reader \*reader) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.*
- **void store** (decaf::io::OutputStream \*out, const std::string &comment) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes this property list (key and element pairs) in this **Properties** (p. 3126) table to the output stream in a format suitable for loading into a **Properties** (p. 3126) table using the load(InputStream) method.*
- **void store** (decaf::io::Writer \*writer, const std::string &comments) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes this property list (key and element pairs) in this **Properties** (p. 3126) table to the output character stream in a format that can be read by the load(Reader) method.*

## Protected Attributes

- **decaf::lang::Pointer< Properties > defaults**  
*Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.*

### 6.650.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 3126) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 3126) instance can contain an **internal** (p. 144) **Properties** (p. 3126) list that contains default values for keys not found in the **Properties** (p. 3126) **List** (p. 2337).

The **Properties** (p. 3126) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since:

1.0

## 6.650.2 Constructor & Destructor Documentation

**6.650.2.1** `decaf::util::Properties::Properties ()`

**6.650.2.2** `decaf::util::Properties::Properties (const Properties & src)`

**6.650.2.3** `virtual decaf::util::Properties::~~Properties ()` [virtual]

## 6.650.3 Member Function Documentation

**6.650.3.1** `void decaf::util::Properties::clear ()`

Clears all properties from the map.

**6.650.3.2** `Properties* decaf::util::Properties::clone () const`

Clones this object.

### Returns:

a replica of this object.

**6.650.3.3** `void decaf::util::Properties::copy (const Properties & source)`

Copies the contents of the given properties object to this one, if the given **Properties** (p. 3126) instance is NULL then this **List** (p. 2337) is not modified.

### Parameters:

*source* The source properties object.

**6.650.3.4** `bool decaf::util::Properties::equals (const Properties & source) const`

Test whether two **Properties** (p. 3126) objects are equivalent. Two **Properties** (p. 3126) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

### Parameters:

*source* The **Properties** (p. 3126) object to compare this instance to.

### Returns:

true if the contents of the two **Properties** (p. 3126) objects are the same.

**6.650.3.5** `std::string decaf::util::Properties::getProperty (const std::string & name, const std::string & default Value) const`

Looks up the value for the given property.



**Parameters:**

*name* The name of the property to be looked up.

*defaultValue* The value to be returned if the given property does not exist.

**Returns:**

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

**6.650.3.6** `const char* decaf::util::Properties::getProperty (const std::string & name) const`

Looks up the value for the given property.

**Parameters:**

*name* The name of the property to be looked up.

**Returns:**

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

**6.650.3.7** `bool decaf::util::Properties::hasProperty (const std::string & name) const`

Check to see if the Property exists in the set.

**Parameters:**

*name* The property name to check for in this properties set.

**Returns:**

true if property exists, false otherwise.

**6.650.3.8** `bool decaf::util::Properties::isEmpty () const`

Returns true if the properties object is empty.

**Returns:**

true if empty

**6.650.3.9** `void decaf::util::Properties::load (decaf::io::Reader * reader) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )`

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format. **Properties** (p. 3126) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character `\`. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII `'#'` or `'!'` as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (`' '`), tab (`'\t'`), and form feed (`'\f'`) to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of  $2n$  contiguous backslashes before a line terminator (or elsewhere) encodes  $n$  backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped `'='`, `':'`, or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\: \=
```

would be the two-character key `":="`. Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is `'='` or `':'`, then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string `""`. Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key `"Truth"` and the associated element value `"Beauty"`:

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is `"fruits"` and the associated element is: `"apple, banana, pear, cantaloupe, watermelon, kiwi, mango"`

Note that a space appears before each `\` so that a space will appear after each comma in the final result; the `\`, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is `"cheeses"` and the associated element is the empty string `""`.

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings

are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, `\`, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence `"\z"` would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence `"\b"` as equivalent to the single character `'b'`.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

**Parameters:**

*reader* The Reader that provides an character stream as input.

**Exceptions:**

*IOException* if there is an error while reading from the stream.

*IllegalArgumentException* if malformed data is found while reading the properties.

*NullPointerException* if the passed stream is Null.

**6.650.3.10** `void decaf::util::Properties::load (decaf::io::InputStream  
* stream) throw ( decaf::io::IOException, de-  
caf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )`

Reads a property list (key and element pairs) from the input byte stream. The input stream is in a simple line-oriented format as specified in `load(Reader)` and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

**Parameters:**

*stream* The stream to read the properties data from.

**Exceptions:**

*IOException* if there is an error while reading from the stream.

*IllegalArgumentException* if malformed data is found while reading the properties.

*NullPointerException* if the passed stream is Null.

**6.650.3.11 Properties& decaf::util::Properties::operator= (const Properties & *src*)**

Assignment Operator.

**Parameters:**

*src* The **Properties** (p. 3126) list to copy to this **List** (p. 2337).

**Returns:**

a reference to this **List** (p. 2337) for use in chaining.

**6.650.3.12 std::vector<std::string> decaf::util::Properties::propertyNames () const**

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

**Returns:**

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

**6.650.3.13 std::string decaf::util::Properties::remove (const std::string & *name*)**

Removes the property with the given name.

**Parameters:**

*name* The name of the property to remove.

**Returns:**

the previous value of the property if set, or empty string.

**6.650.3.14 std::string decaf::util::Properties::setProperty (const std::string & *name*, const std::string & *value*)**

Sets the value for a given property. If the property already exists, overwrites the value.

**Parameters:**

*name* The name of the value to be written.

*value* The value to be written.

**Returns:**

the old value of the property or empty string if not set.

**6.650.3.15 std::size\_t decaf::util::Properties::size () const****Returns:**

The number of **Properties** (p. 3126) in this **Properties** (p. 3126) Object.

**6.650.3.16** `void decaf::util::Properties::store (decaf::io::Writer * writer,  
const std::string & comments) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException )`

Writes this property list (key and element pairs) in this **Properties** (p. 3126) table to the output character stream in a format that can be read by the load(Reader) method. **Properties** (p. 3126) from the defaults table of this **Properties** (p. 3126) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (

'), a carriage return ("), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of **Date** (p. 1665) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 3126) table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

#### Parameters:

*writer* The Writer instance to use to output the properties.

*comments* A description of these properties that is written before writing the properties.

#### Exceptions:

*IOException* if there is an error while writing from the stream.

*NullPointerException* if the passed stream is Null.

**6.650.3.17** `void decaf::util::Properties::store (decaf::io::OutputStream * out,  
const std::string & comment) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException )`

Writes this property list (key and element pairs) in this **Properties** (p. 3126) table to the output stream in a format suitable for loading into a **Properties** (p. 3126) table using the load(InputStream) method. **Properties** (p. 3126) from the defaults table of this **Properties** (p. 3126) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.

- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

**Parameters:**

*out* The OutputStream instance to write the properties to.

*comment* A description of these properties that is written to the output stream.

**Exceptions:**

*IOException* if there is an error while writing from the stream.

*NullPointerException* if the passed stream is Null.

**6.650.3.18** `std::vector< std::pair< std::string, std::string > >  
decaf::util::Properties::toArray () const`

Method that serializes the contents of the property map to an array.

**Returns:**

list of pairs where the first is the name and the second is the value.

**6.650.3.19** `std::string decaf::util::Properties::toString () const`

Formats the contents of the **Properties** (p. 3126) Object into a string that can be logged, etc.

**Returns:**

string value of this object.

## 6.650.4 Field Documentation

**6.650.4.1** `decaf::lang::Pointer<Properties> decaf::util::Properties::defaults  
[protected]`

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

## 6.651 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 3126).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

### Public Member Functions

- virtual **~PropertiesChangeListener** ()
- virtual void **onPropertiesReset** ()=0  
*Indicates that the **Properties** (p. 3126) have all been reset and should be considered to be back to their default values.*
- virtual void **onPropertyChanged** (const std::string &name, const std::string &oldValue, const std::string &newValue)=0  
*Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.*

### 6.651.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 3126).

Since:

1.0

### 6.651.2 Constructor & Destructor Documentation

- 6.651.2.1** virtual  
**decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener**  
() [inline, virtual]

### 6.651.3 Member Function Documentation

- 6.651.3.1** virtual void **decaf::util::logging::PropertiesChangeListener::onPropertiesReset** () [pure virtual]

Indicates that the **Properties** (p. 3126) have all been reset and should be considered to be back to their default values.

- 6.651.3.2** virtual void **decaf::util::logging::PropertiesChangeListener::onPropertyChanged** (const std::string & *name*, const std::string & *oldValue*, const std::string & *newValue*) [pure virtual]

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

**Parameters:**

*name* The name of the Property that changed.

*oldValue* The old Value of the Property.

*newValue* The new Value of the Property.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**



## 6.652 decaf::net::ProtocolException Class Reference

#include <src/main/decaf/net/ProtocolException.h> Inheritance diagram for decaf::net::ProtocolException:

### Public Member Functions

- **ProtocolException** () throw ()  
*Default Constructor.*
- **ProtocolException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ProtocolException** (const **ProtocolException** &ex) throw ()  
*Copy Constructor.*
- **ProtocolException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ProtocolException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ProtocolException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ProtocolException** \* clone () const  
*Clones this exception.*
- virtual ~**ProtocolException** () throw ()

### 6.652.1 Constructor & Destructor Documentation

#### 6.652.1.1 decaf::net::ProtocolException::ProtocolException () throw () [inline]

Default Constructor.

#### 6.652.1.2 decaf::net::ProtocolException::ProtocolException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

### 6.652.1.3 decaf::net::ProtocolException::ProtocolException (const ProtocolException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

### 6.652.1.4 decaf::net::ProtocolException::ProtocolException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.652.1.5 decaf::net::ProtocolException::ProtocolException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.652.1.6 decaf::net::ProtocolException::ProtocolException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.652.1.7** virtual decaf::net::ProtocolException::~~ProtocolException () throw ()  
[inline, virtual]

## 6.652.2 Member Function Documentation

**6.652.2.1** virtual ProtocolException\* decaf::net::ProtocolException::clone () const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ProtocolException.h**

## 6.653 decaf::security::PublicKey Class Reference

A public key.

`#include <src/main/decaf/security/PublicKey.h>`  
Inheritance diagram for decaf::security::PublicKey:

### Public Member Functions

- virtual `~PublicKey()`

#### 6.653.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

#### 6.653.2 Constructor & Destructor Documentation

##### 6.653.2.1 virtual `decaf::security::PublicKey::~~PublicKey()` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

## 6.654 decaf::io::PushbackInputStream Class Reference

A **PushbackInputStream** (p. 3141) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

#include <src/main/decaf/io/PushbackInputStream.h> Inheritance diagram for decaf::io::PushbackInputStream:

### Public Member Functions

- **PushbackInputStream** (**InputStream** \*stream, bool **own**=false)  
*Creates a **PushbackInputStream** (p. 3141) and saves its argument, the input stream in, for later use.*
- **PushbackInputStream** (**InputStream** \*stream, int bufSize, bool **own**=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **PushbackInputStream** (p. 3141) and saves its argument, the input stream in, for later use.*
- virtual ~**PushbackInputStream** ()
- void **unread** (unsigned char value) throw ( decaf::io::IOException )  
*Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.*
- void **unread** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*
- void **unread** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data avaiable. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.*  
**Returns:**  
*the number of bytes available on this input stream.*  
**Exceptions:**  
*IOException (p. 2142) if an I/O error occurs.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards  $n$  bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before  $n$  bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until  $num$  bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters:**

***num** The number of bytes to skip.*

**Returns:**

*total bytes skipped*

**Exceptions:**

***IOException** (p. 2142) if an I/O error occurs.*

***UnsupportedOperationException** if the concrete stream class does not support skipping bytes.*

- virtual void **mark** (int readLimit)

*Does nothing except throw an **IOException** (p. 2142).*

- virtual void **reset** () throw ( decaf::io::IOException )

*Does nothing except throw an **IOException** (p. 2142).*

- virtual bool **markSupported** () const

*Does nothing except throw an **IOException** (p. 2142).*

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsExpection, decaf::lang::exceptions::NullPointerException )

### 6.654.1 Detailed Description

A **PushbackInputStream** (p. 3141) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte. This is useful in situations where it is convenient for a fragment of **code** (p. 1183) to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the **code** (p. 1183) fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

**Since:**

1.0

## 6.654.2 Constructor & Destructor Documentation

### 6.654.2.1 decaf::io::PushbackInputStream::PushbackInputStream (InputStream \* *stream*, bool *own* = false)

Creates a **PushbackInputStream** (p. 3141) and saves its argument, the input stream in, for later use. Initially, there is no pushed-back byte.

#### Parameters:

*stream* The **InputStream** (p. 2043) instance to wrap.

*Boolean* value indicating if this **FilterInputStream** (p. 1894) owns the wrapped stream.

### 6.654.2.2 decaf::io::PushbackInputStream::PushbackInputStream (InputStream \* *stream*, int *bufSize*, bool *own* = false) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **PushbackInputStream** (p. 3141) and saves its argument, the input stream in, for later use. Initially, there is no pushed-back byte.

#### Parameters:

*stream* The **InputStream** (p. 2043) instance to wrap.

*bufSize* The number of byte to allocate for pushback into this stream.

*Boolean* value indicating if this **FilterInputStream** (p. 1894) owns the wrapped stream.

#### Exceptions:

*IllegalArgumentException* if the *bufSize* argument is < zero.

### 6.654.2.3 virtual decaf::io::PushbackInputStream::~~PushbackInputStream () [virtual]

## 6.654.3 Member Function Documentation

### 6.654.3.1 virtual int decaf::io::PushbackInputStream::available () const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns:

the number of bytes available on this input stream.

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to `available`.

Reimplemented from `decaf::io::FilterInputStream` (p.1896).

**6.654.3.2** `virtual int decaf::io::PushbackInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1897).

**6.654.3.3** `virtual int decaf::io::PushbackInputStream::doReadByte () throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p.1897).

**6.654.3.4** `virtual void decaf::io::PushbackInputStream::mark (int readLimit)` [virtual]

Does nothing except throw an **IOException** (p.2142). Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters:

***readLimit*** The max bytes read before marked position is invalid.

Reimplemented from `decaf::io::FilterInputStream` (p.1897).

**6.654.3.5** `virtual bool decaf::io::PushbackInputStream::markSupported () const` [inline, virtual]

Does nothing except throw an **IOException** (p.2142). Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns:

true if this stream instance supports marks

Reimplemented from `decaf::io::FilterInputStream` (p.1898).



### 6.654.3.6 virtual void decaf::io::PushbackInputStream::reset () throw ( decaf::io::IOException ) [virtual]

Does nothing except throw an **IOException** (p. 2142). Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2142) might be thrown. \* If such an **IOException** (p. 2142) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2142). \* If an **IOException** (p. 2142) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2142).

#### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1898).

### 6.654.3.7 virtual long long decaf::io::PushbackInputStream::skip (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2043) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

**num** The number of bytes to skip.

#### Returns:

total bytes skipped

#### Exceptions:

**IOException** (p. 2142) if an I/O error occurs.

**UnsupportedOperationException** if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p.1899).

**6.654.3.8** `void decaf::io::PushbackInputStream::unread (const unsigned char *  
buffer, int size, int offset, int length) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

**Parameters:**

*buffer* The bytes to copy to the front of push back buffer.

*size* The size of the array to be copied.

*offset* The position in the buffer to start copying from.

*length* The number of bytes to push back from the passed buffer.

**Exceptions:**

*NullPointerException* if the buffer passed is NULL.

*IndexOutOfBoundsException* if the offset + length is greater than the buffer size.

*IOException* (p. 2142) if there is not enough space in the pushback buffer or this stream has already been closed.

**6.654.3.9** `void decaf::io::PushbackInputStream::unread (const unsigned  
char * buffer, int size) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

**Parameters:**

*buffer* The bytes to copy to the front of push back buffer.

*size* The size of the array to be copied.

**Exceptions:**

*NullPointerException* if the buffer passed is NULL.

*IndexOutOfBoundsException* if the size value given is negative.

*IOException* (p. 2142) if there is not enough space in the pushback buffer or this stream has already been closed.

**6.654.3.10**    `void decaf::io::PushbackInputStream::unread (unsigned char value)  
                  throw ( decaf::io::IOException )`

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

**Parameters:**

*value* The byte that is to be placed at the front of the push back buffer.

**Exceptions:**

*IOException* (p. 2142) if there is not enough space in the pushback buffer or this stream has already been closed.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/`**PushbackInputStream.h**

## 6.655 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

#include <src/main/cms/Queue.h> Inheritance diagram for cms::Queue:

### Public Member Functions

- virtual `~Queue ()`
- virtual `std::string getQueueName () const =0 throw ( CMSEException )`  
*Gets the name of this queue.*

### 6.655.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 3148) are sent to a Single Subscriber on that **Queue** (p. 3148) **Destination** (p. 1723). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2534) in a **Queue** (p. 3148) is not defined by the CMS API, consult your Provider documentation for this information.

**Since:**

1.0

### 6.655.2 Constructor & Destructor Documentation

**6.655.2.1** virtual `cms::Queue::~Queue ()` [inline, virtual]

### 6.655.3 Member Function Documentation

**6.655.3.1** virtual `std::string cms::Queue::getQueueName () const throw ( CMSEException )` [pure virtual]

Gets the name of this queue.

**Returns:**

The queue name.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQQueue` (p. 486).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

## 6.656 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

#include <src/main/decaf/util/Queue.h> Inheritance diagram for decaf::util::Queue< E >:

### Public Member Functions

- virtual **~Queue** ()
- virtual bool **offer** (const E &value)=0 throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the queue provided that the condition allows such an operation.*
- virtual bool **poll** (E &result)=0  
*Gets and removes the element in the head of the queue.*
- virtual E **remove** ()=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const =0  
*Gets but not removes the element in the head of the queue.*
- virtual E **element** () const =0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets but not removes the element in the head of the queue.*

### 6.656.1 Detailed Description

**template<typename E> class decaf::util::Queue< E >**

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

**Queue** (p.3149) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 3149) interface the methods of this class cannot return null to indicate that a **Queue** (p. 3149) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 3149) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 3149) must be *assignable* in order to utilize these methods.

**Since:**

1.0

## 6.656.2 Constructor & Destructor Documentation

**6.656.2.1** `template<typename E > virtual decaf::util::Queue< E >::~~Queue ()`  
`[inline, virtual]`

## 6.656.3 Member Function Documentation

**6.656.3.1** `template<typename E > virtual E decaf::util::Queue< E >::element ()`  
`const throw ( decaf::lang::exceptions::NoSuchElementException ) [pure`  
`virtual]`

Gets but not removes the element in the head of the queue. Throws a `NoSuchElementException` if there is no element in the queue.

### Returns:

the element in the head of the queue.

### Exceptions:

***NoSuchElementException*** if there is no element in the queue.

Implemented in `decaf::util::AbstractQueue< E >` (p. 196).

**6.656.3.2** `template<typename E > virtual bool decaf::util::Queue< E >::offer`  
`(const E & value) throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation. The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

### Parameters:

*value* the specified element to insert into the queue.

### Returns:

true if the operation succeeds and false if it fails.

### Exceptions:

***NullPointerException*** if the **Queue** (p. 3149) implementation does not allow Null values to be inserted into the **Queue** (p. 3149).

***IllegalArgumentException*** if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3721), and `decaf::util::PriorityQueue< E >` (p. 3032).

Referenced by `decaf::util::AbstractQueue< E >::add()`.

**6.656.3.3** `template<typename E> virtual bool decaf::util::Queue< E >::peek (E & result) const` [pure virtual]

Gets but not removes the element in the head of the queue. The result if successful is assigned to the result parameter.

**Parameters:**

*result* Reference to an instance of the contained type to assigned the removed value to.

**Returns:**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in `decaf::util::PriorityQueue< E >` (p. 3033).

Referenced by `decaf::util::AbstractQueue< E >::element()`.

**6.656.3.4** `template<typename E> virtual bool decaf::util::Queue< E >::poll (E & result)` [pure virtual]

Gets and removes the element in the head of the queue. If the operation succeeds the value of the element at the head of the **Queue** (p. 3149) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

**Parameters:**

*result* Reference to an instance of the contained type to assigned the removed value to.

**Returns:**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3722), and `decaf::util::PriorityQueue< E >` (p. 3033).

Referenced by `decaf::util::AbstractQueue< E >::clear()`, and `decaf::util::AbstractQueue< E >::remove()`.

**6.656.3.5** `template<typename E> virtual E decaf::util::Queue< E >::remove () throw ( decaf::lang::exceptions::NoSuchElementException )` [pure virtual]

Gets and removes the element in the head of the queue. Throws a `NoSuchElementException` if there is no element in the queue.

**Returns:**

the element in the head of the queue.

**Exceptions:**

*NoSuchElementException* if there is no element in the queue.

Implemented in `decaf::util::AbstractQueue< E >` (p. 196), and `decaf::util::PriorityQueue< E >` (p. 3034).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Queue.h`



## 6.657 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p.3148) without removing them.

#include <src/main/cms/QueueBrowser.h> Inheritance diagram for cms::QueueBrowser:

### Public Member Functions

- virtual **~QueueBrowser** ()
- virtual const **Queue** \* **getQueue** () const =0 throw ( cms::CMSEException )
- virtual std::string **getMessageSelector** () const =0 throw ( cms::CMSEException )
- virtual **cms::MessageEnumeration** \* **getEnumeration** ()=0 throw ( cms::CMSEException )

*Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p.3148) in the order that a client would receive them.*

### 6.657.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p.3148) without removing them. To browse the contents of the **Queue** (p.3148) the client calls the **getEnumeration** method to retrieve a new instance of a **Queue** (p.3148) Enumerator. The client then calls the **hasMoreMessages** method of the Enumeration, if it returns true the client can safely call the **nextMessage** method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration() (p.3153);
```

```
while( enumeration->hasMoreMessages() ) { cms::Message (p.2534)* message = enumeration->nextMessage();
```

```
// ... Do something with the Message (p.2534).
```

```
delete message; }
```

Since:

1.1

### 6.657.2 Constructor & Destructor Documentation

**6.657.2.1** virtual cms::QueueBrowser::~~QueueBrowser () [inline, virtual]

### 6.657.3 Member Function Documentation

**6.657.3.1** virtual cms::MessageEnumeration\* cms::QueueBrowser::getEnumeration () throw ( cms::CMSEException ) [pure virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p.3148) in the order that a client would receive them. The pointer returned is owned by the browser and should not be deleted by the client application.

**Returns:**

a pointer to a **Queue** (p. 3148) Enumeration, this Pointer is owned by the **QueueBrowser** (p. 3153) and should not be deleted by the client.

**Exceptions:**

*CMSEException* (p. 1160) if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 488).

**6.657.3.2** `virtual std::string cms::QueueBrowser::getMessageSelector () const  
throw ( cms::CMSEException ) [pure virtual]`

**Returns:**

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

**Exceptions:**

*CMSEException* (p. 1160) if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 488).

**6.657.3.3** `virtual const Queue* cms::QueueBrowser::getQueue () const throw (  
cms::CMSEException ) [pure virtual]`

**Returns:**

the **Queue** (p. 3148) that this browser is listening on.

**Exceptions:**

*CMSEException* (p. 1160) if an internal error occurs.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 489).

The documentation for this class was generated from the following file:

- `src/main/cms/QueueBrowser.h`

## 6.658 decaf::util::Random Class Reference

**Random** (p. 3155) Value Generator which is used to generate a stream of pseudorandom numbers.

#include <src/main/decaf/util/Random.h> Inheritance diagram for decaf::util::Random:

### Public Member Functions

- **Random** ()  
*Construct a random generator with the current time of day in milliseconds as the initial state.*
- **Random** (unsigned long long seed)  
*Construct a random generator with the given **seed** as the initial state.*
- bool **nextBoolean** ()  
*Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.*
- double **nextDouble** ()  
*Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.*
- float **nextFloat** ()  
*Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.*
- double **nextGaussian** ()  
*Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.*
- int **nextInt** ()  
*Generates a uniformly distributed 32-bit **int** value from the this random number sequence.*
- int **nextInt** (int n)  
*Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of **n** (exclusively).*
- long long **nextLong** ()  
*Generates a uniformly distributed 64-bit **int** value from the this random number sequence.*
- virtual void **nextBytes** (std::vector< unsigned char > &buf)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*
- virtual void **nextBytes** (unsigned char \*buf, int size)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*
- virtual void **setSeed** (unsigned long long seed)  
*Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.*

## Protected Member Functions

- virtual int **next** (int bits)

*Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E.*

### 6.658.1 Detailed Description

**Random** (p. 3155) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 3155) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since:

1.0

### 6.658.2 Constructor & Destructor Documentation

#### 6.658.2.1 decaf::util::Random::Random ()

Construct a random generator with the current time of day in milliseconds as the initial state.

See also:

**setSeed** (p. 3159)

#### 6.658.2.2 decaf::util::Random::Random (unsigned long long *seed*)

Construct a random generator with the given **seed** as the initial state.

Parameters:

*seed* the seed that will determine the initial state of this random number generator

See also:

**setSeed** (p. 3159)

### 6.658.3 Member Function Documentation

#### 6.658.3.1 virtual int decaf::util::Random::next (int *bits*) [protected, virtual]

Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns:

int a pseudo-random generated int number

Parameters:

*bits* number of bits of the returned value

See also:

**nextBytes** (p. 3157)  
**nextDouble** (p. 3158)  
**nextFloat** (p. 3158)  
**nextInt()** (p. 3159)  
**nextInt(int)** (p. 3158)  
**nextGaussian** (p. 3158)  
**nextLong** (p. 3159)

Reimplemented in **decaf::security::SecureRandom** (p. 3333).

### 6.658.3.2 bool decaf::util::Random::nextBoolean ()

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

**Returns:**

boolean a pseudo-random, uniformly distributed boolean value

### 6.658.3.3 virtual void decaf::util::Random::nextBytes (unsigned char \* *buf*, int *size*) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

**Parameters:**

*buf* non-null array to contain the new random bytes

See also:

**next** (p. 3156)

**Exceptions:**

*NullPointerException* if *buff* is NULL

*IllegalArgumentException* if *size* is negative

Reimplemented in **decaf::security::SecureRandom** (p. 3334).

### 6.658.3.4 virtual void decaf::util::Random::nextBytes (std::vector< unsigned char > & *buf*) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

**Parameters:**

*buf* non-null array to contain the new random bytes

See also:

**next** (p. 3156)

Reimplemented in **decaf::security::SecureRandom** (p. 3334).

**6.658.3.5 double decaf::util::Random::nextDouble ()**

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

**Returns:**

double

**See also:**

**nextFloat** (p. 3158)

**6.658.3.6 float decaf::util::Random::nextFloat ()**

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

**Returns:**

float a random float number between 0.0 and 1.0

**See also:**

**nextDouble** (p. 3158)

**6.658.3.7 double decaf::util::Random::nextGaussian ()**

Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G. E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

**Returns:**

double

**See also:**

**nextDouble** (p. 3158)

**6.658.3.8 int decaf::util::Random::nextInt (int n)**

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of **n** (exclusively).

**Parameters:**

**n** The int value that defines the max value of the return.

**Returns:**

the next pseudo random int value.

**Exceptions:**

***IllegalArgumentException*** if **n** is less than or equal to zero.

### 6.658.3.9 int decaf::util::Random::nextInt ()

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

**Returns:**

`int` uniformly distributed `int` value

**See also:**

`next` (p. 3156)  
`nextLong` (p. 3159)

### 6.658.3.10 long long decaf::util::Random::nextLong ()

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

**Returns:**

64-bit `int` random number

**See also:**

`next` (p. 3156)  
`nextInt()` (p. 3159)  
`nextInt(int)` (p. 3158)

### 6.658.3.11 virtual void decaf::util::Random::setSeed (unsigned long long *seed*) [virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

**Parameters:**

*seed* the seed that alters the state of the random number generator

**See also:**

`next` (p. 3156)  
`Random()` (p. 3156)  
`Random(long)`

Reimplemented in `decaf::security::SecureRandom` (p. 3335).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Random.h`

## 6.659 decaf::lang::Readable Class Reference

A **Readable** (p. 3160) is a source of characters.

#include <src/main/decaf/lang/Readable.h>Inheritance diagram for decaf::lang::Readable:

### Public Member Functions

- virtual **~Readable** ()
- virtual int **read** (**decaf::nio::CharBuffer** \*charBuffer)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

### 6.659.1 Detailed Description

A **Readable** (p. 3160) is a source of characters. Characters from a **Readable** (p. 3160) are made available to callers of the read method via a CharBuffer.

Since:

1.0

### 6.659.2 Constructor & Destructor Documentation

**6.659.2.1** virtual decaf::lang::Readable::~~Readable () [inline, virtual]

### 6.659.3 Member Function Documentation

**6.659.3.1** virtual int decaf::lang::Readable::read (decaf::nio::CharBuffer \* *charBuffer*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [pure virtual]

Attempts to read characters into the specified character buffer. The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters:

*charBuffer* The Buffer to read Characters into.

Returns:

The number of char values added to the buffer, or -1 if this source of characters is at its end

Exceptions:

*IOException* - if an I/O error occurs

*NullPointerException* - if buffer is NULL.



*ReadOnlyBufferException* - if charBuffer is a read only buffer

Implemented in **decaf::io::Reader** (p. 3165).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Readable.h**

## 6.660 activemq::transport::inactivity::ReadChecker Class Reference

Runnable class that is used by the {}.

#include <src/main/activemq/transport/inactivity/ReadChecker.h>Inheritance diagram for activemq::transport::inactivity::ReadChecker:

### Public Member Functions

- **ReadChecker** (**InactivityMonitor** \*parent)
- virtual ~**ReadChecker** ()
- virtual void **run** ()

*Run method - called by the Thread class in the context of the thread.*

### 6.660.1 Detailed Description

Runnable class that is used by the {}.

See also:

**InactivityMonitor** (p. 2004)} class the check for timeouts related to **transport** (p. 97) reads.

Since:

3.1

### 6.660.2 Constructor & Destructor Documentation

**6.660.2.1** **activemq::transport::inactivity::ReadChecker::ReadChecker** (**InactivityMonitor** \* *parent*)

**6.660.2.2** **virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker** ()  
[virtual]

### 6.660.3 Member Function Documentation

**6.660.3.1** **virtual void activemq::transport::inactivity::ReadChecker::run** ()  
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3325).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**ReadChecker.h**

## 6.661 decaf::io::Reader Class Reference

#include <src/main/decaf/io/Reader.h> Inheritance diagram for decaf::io::Reader:

### Public Member Functions

- virtual **~Reader** ()
- virtual void **mark** (int readAheadLimit) throw ( decaf::io::IOException )  
*Marks the present position in the stream.*
- virtual bool **markSupported** () const  
*Tells whether this stream supports the **mark()** (p. 3165) operation.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Resets the stream.*
- virtual long long **skip** (long long count) throw ( decaf::io::IOException )  
*Skips characters.*
- virtual int **read** (std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Reads characters into an array.*
- virtual int **read** (char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reads characters into an array, the method will attempt to read as much data as the size of the array.*
- virtual int **read** (char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Reads characters into a portion of an array.*
- virtual int **read** () throw ( decaf::io::IOException )  
*Reads a single character.*
- virtual int **read** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )  
*Attempts to read characters into the specified character buffer.*

### Protected Member Functions

- **Reader** ()

- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ).*
- virtual int **doReadVector** (std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Override this method to customize the functionality of the method read( std::vector<char>& buffer ) (p. 3167).*
- virtual int **doReadArray** (char \*buffer, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Override this method to customize the functionality of the method read( char\* buffer, std::size\_t length ).*
- virtual int **doReadChar** () throw ( decaf::io::IOException )  
*Override this method to customize the functionality of the method read() (p. 3166).*
- virtual int **doReadCharBuffer** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )  
*Override this method to customize the functionality of the method read( CharBuffer\* charBuffer ).*

## 6.661.1 Constructor & Destructor Documentation

6.661.1.1 decaf::io::Reader::Reader () [protected]

6.661.1.2 virtual decaf::io::Reader::~~Reader () [virtual]

## 6.661.2 Member Function Documentation

6.661.2.1 virtual int decaf::io::Reader::doReadArray (char \* buffer, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Override this method to customize the functionality of the method read( char\* buffer, std::size\_t length ).

6.661.2.2 virtual int decaf::io::Reader::doReadArrayBounded (char \* buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, pure virtual]

Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ). All subclasses must override this method to provide the basic **Reader** (p. 3163) functionality.

Implemented in **decaf::io::InputStreamReader** (p. 2054).

**6.661.2.3** `virtual int decaf::io::Reader::doReadChar () throw ( decaf::io::IOException ) [protected, virtual]`

Override this method to customize the functionality of the method `read()` (p. 3166).

**6.661.2.4** `virtual int decaf::io::Reader::doReadCharBuffer (decaf::nio::CharBuffer * charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [protected, virtual]`

Override this method to customize the functionality of the method `read( CharBuffer* charBuffer )`.

**6.661.2.5** `virtual int decaf::io::Reader::doReadVector (std::vector< char > & buffer) throw ( decaf::io::IOException ) [protected, virtual]`

Override this method to customize the functionality of the method `read( std::vector<char>& buffer )` (p. 3167).

**6.661.2.6** `virtual void decaf::io::Reader::mark (int readAheadLimit) throw ( decaf::io::IOException ) [virtual]`

Marks the present position in the stream. Subsequent calls to `reset()` (p. 3168) will attempt to reposition the stream to this point. Not all character-input streams support the `mark()` (p. 3165) operation.

#### Parameters:

*readAheadLimit* Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs, or the stream does not support mark.

**6.661.2.7** `virtual bool decaf::io::Reader::markSupported () const [inline, virtual]`

Tells whether this stream supports the `mark()` (p. 3165) operation. The default implementation always returns false. Subclasses should override this method.

#### Returns:

true if and only if this stream supports the mark operation.

**6.661.2.8** `virtual int decaf::io::Reader::read (decaf::nio::CharBuffer * charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Attempts to read characters into the specified character buffer. The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or

rewinding of the buffer is performed.

**Parameters:**

*charBuffer* The Buffer to read Characters into.

**Returns:**

The number of char values added to the buffer, or -1 if this source of characters is at its end

**Exceptions:**

*IOException* (p. 2142) - if an I/O error occurs

*NullPointerException* - if buffer is NULL.

*ReadOnlyBufferException* - if charBuffer is a read only buffer

Implements **decaf::lang::Readable** (p. 3160).

**6.661.2.9 virtual int decaf::io::Reader::read () throw ( decaf::io::IOException )**  
[virtual]

Reads a single character. This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

**Returns:**

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

**Exceptions:**

*IOException* (p. 2142) thrown if an I/O error occurs.

**6.661.2.10 virtual int decaf::io::Reader::read (char \* buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )** [virtual]

Reads characters into a portion of an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

**Parameters:**

*buffer* The target char buffer.

*size* The size in bytes of the target buffer.

*offset* The position in the buffer to start filling.

*length* The maximum number of bytes to read.

**Returns:**

The number of bytes read or -1 if the end of stream is reached.

**Exceptions:**

*IOException* (p. 2142) thrown if an I/O error occurs.

*NullPointerException* if buffer is NULL.

*IndexOutOfBoundsException* if the offset + length is greater than the array size.

**6.661.2.11** `virtual int decaf::io::Reader::read (char * buffer, int size) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`  
[virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

**Parameters:**

*buffer* The target char buffer.

*size* The size in bytes of the target buffer.

**Returns:**

The number of bytes read or -1 if the end of stream is reached.

**Exceptions:**

*IOException* (p. 2142) thrown if an I/O error occurs.

*NullPointerException* if buffer is NULL.

**6.661.2.12** `virtual int decaf::io::Reader::read (std::vector< char > & buffer) throw (decaf::io::IOException )` [virtual]

Reads characters into an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

**Parameters:**

*buffer* The buffer to read characters into.

**Returns:**

The number of characters read, or -1 if the end of the stream has been reached

**Exceptions:**

*IOException* (p. 2142) thrown if an I/O error occurs.

**6.661.2.13** `virtual bool decaf::io::Reader::ready () const throw (decaf::io::IOException )` [virtual]

Tells whether this stream is ready to be read.

**Returns:**

True if the next `read()` (p. 3166) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented in `decaf::io::InputStreamReader` (p. 2054).

**6.661.2.14 virtual void decaf::io::Reader::reset () throw ( decaf::io::IOException )  
[virtual]**

Resets the stream. If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the `reset()` (p. 3168) operation, and some support `reset()` (p. 3168) without supporting `mark()` (p. 3165).

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

**6.661.2.15 virtual long long decaf::io::Reader::skip (long long count) throw ( decaf::io::IOException ) [virtual]**

Skips characters. This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

**Parameters:**

*count* The number of character to skip.

**Returns:**

the number of Character actually skipped.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Reader.h`



## 6.662 decaf::nio::ReadOnlyBufferException Class Reference

#include <src/main/decaf/nio/ReadOnlyBufferException.h> Inheritance diagram for decaf::nio::ReadOnlyBufferException:

### Public Member Functions

- **ReadOnlyBufferException** () throw ()  
*Default Constructor.*
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const ReadOnlyBufferException &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ReadOnlyBufferException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **ReadOnlyBufferException \* clone** () const  
*Clones this exception.*
- virtual ~**ReadOnlyBufferException** () throw ()

### 6.662.1 Constructor & Destructor Documentation

#### 6.662.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException () throw () [inline]

Default Constructor.

#### 6.662.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

### 6.662.1.3 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & ex) throw () [inline]`

Copy Constructor.

#### Parameters:

*ex* the exception to copy, which is an instance of this type

### 6.662.1.4 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.662.1.5 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const std::exception * cause) throw () [inline]`

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.662.1.6 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.662.1.7** virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException  
( ) throw ( ) [inline, virtual]

## 6.662.2 Member Function Documentation

**6.662.2.1** virtual ReadOnlyBufferException\* de-  
caf::nio::ReadOnlyBufferException::clone ( ) const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p. 3918).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ReadOnlyBufferException.h**

## 6.663 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 3172) maintains a pair of associated **locks** (p. 174), one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

### Public Member Functions

- virtual **~ReadWriteLock** ()
- virtual **Lock & readLock** ()=0  
*Returns the lock used for reading.*
- virtual **Lock & writeLock** ()=0  
*Returns the lock used for writing.*

### 6.663.1 Detailed Description

A **ReadWriteLock** (p. 3172) maintains a pair of associated **locks** (p. 174), one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 3172) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 2377) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of **code** (p. 1183). Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

\* Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible. \* Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. \* Determining whether the **locks** (p. 174) are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? \* Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

**Since:**

1.0

**6.663.2 Constructor & Destructor Documentation**

**6.663.2.1** `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock ()`  
[inline, virtual]

**6.663.3 Member Function Documentation**

**6.663.3.1** `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock ()`  
[pure virtual]

Returns the lock used for reading.

**Returns:**

the lock used for reading.

**6.663.3.2** `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock ()`  
[pure virtual]

Returns the lock used for writing.

**Returns:**

the lock used for writing.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReadWriteLock.h`

## 6.664 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

### Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** \*parent, **cms::Destination** \*destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session) throw (cms::CMSException)

*Execute any number of operations against the supplied CMS session.*

- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session AMQCPP\_UNUSED) throw ( cms::CMSException )
- **cms::Message** \* **getMessage** ()

### Protected Member Functions

- **ReceiveExecutor** (const **ReceiveExecutor** &)
- **ReceiveExecutor** & **operator=** (const **ReceiveExecutor** &)

### Protected Attributes

- **cms::Destination** \* destination
- std::string selector
- bool noLocal
- **cms::Message** \* message
- **CmsTemplate** \* parent

## 6.664.1 Constructor & Destructor Documentation

- 6.664.1.1** `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor (const ReceiveExecutor &) [inline, protected]`
- 6.664.1.2** `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor (CmsTemplate * parent, cms::Destination * destination, const std::string & selector, bool noLocal) [inline]`
- 6.664.1.3** `virtual  
activemq::cmsutil::CmsTemplate::ReceiveExecutor::~~ReceiveExecutor ()  
[inline, virtual]`

## 6.664.2 Member Function Documentation

- 6.664.2.1** `virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms (cms::Session * session) throw (cms::CMSException) [virtual]`

Execute any number of operations against the supplied CMS session.

### Parameters:

*session* the CMS Session

### Exceptions:

*cms::CMSException* (p. 1160) if thrown by CMS API methods

Implements `activemq::cmsutil::SessionCallback` (p. 3378).

- 6.664.2.2** `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session AMQCPP_UNUSED) throw (cms::CMSException) [inline, virtual]`
- 6.664.2.3** `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage () [inline]`
- 6.664.2.4** `ReceiveExecutor& activemq::cmsutil::CmsTemplate::ReceiveExecutor::operator= (const ReceiveExecutor &) [inline, protected]`

Reimplemented in `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor` (p. 3279).

### 6.664.3 Field Documentation

- 6.664.3.1 `cms::Destination*` `activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination`  
[protected]
- 6.664.3.2 `cms::Message*` `activemq::cmsutil::CmsTemplate::ReceiveExecutor::message`  
[protected]
- 6.664.3.3 `bool` `activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal`  
[protected]
- 6.664.3.4 `CmsTemplate*` `activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent`  
[protected]
- 6.664.3.5 `std::string` `activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`



## 6.665 activemq::core::RedeliveryPolicy Class Reference

Interface for a **RedeliveryPolicy** (p. 3177) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

#include <src/main/activemq/core/RedeliveryPolicy.h> Inheritance diagram for activemq::core::RedeliveryPolicy:

### Public Member Functions

- virtual **~RedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () const =0
- virtual void **setBackOffMultiplier** (double value)=0  
*Sets the Back-Off Multiplier for Message Redelivery.*
- virtual short **getCollisionAvoidancePercent** () const =0
- virtual void **setCollisionAvoidancePercent** (short value)=0
- virtual long long **getInitialRedeliveryDelay** () const =0  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void **setInitialRedeliveryDelay** (long long value)=0  
*Sets the initial time that redelivery will be delayed.*
- virtual int **getMaximumRedeliveries** () const =0  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void **setMaximumRedeliveries** (int maximumRedeliveries)=0  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual long long **getRedeliveryDelay** (long long previousDelay)=0  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual bool **isUseCollisionAvoidance** () const =0
- virtual void **setUseCollisionAvoidance** (bool value)=0
- virtual bool **isUseExponentialBackOff** () const =0
- virtual void **setUseExponentialBackOff** (bool value)=0
- virtual **RedeliveryPolicy** \* **clone** () const =0  
*Create a copy of this Policy and return it.*
- virtual void **configure** (const **decaf::util::Properties** &properties)  
*Checks the supplied properties object for properties matching the configurable settings of this class.*

### Static Public Attributes

- static const long long **NO\_MAXIMUM\_REDELIVERIES**

## Protected Member Functions

- **RedeliveryPolicy** ()

### 6.665.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 3177) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

**Since:**

3.2.0

### 6.665.2 Constructor & Destructor Documentation

**6.665.2.1** `activemq::core::RedeliveryPolicy::RedeliveryPolicy ()` [protected]

**6.665.2.2** `virtual activemq::core::RedeliveryPolicy::~~RedeliveryPolicy ()` [virtual]

### 6.665.3 Member Function Documentation

**6.665.3.1** `virtual RedeliveryPolicy* activemq::core::RedeliveryPolicy::clone () const`  
[pure virtual]

Create a copy of this Policy and return it.

**Returns:**

pointer to a new **RedeliveryPolicy** (p. 3177) that is a copy of this one.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1678).

**6.665.3.2** `virtual void activemq::core::RedeliveryPolicy::configure (const decaf::util::Properties & properties)` [virtual]

Checks the supplied properties object for properties matching the configurable settings of this class. The default implementation looks for properties named with the prefix `cms.RedeliveryPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.RedeliveryPolicy.useExponentialBackOff` will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

**Parameters:**

*properties* The Properties object used to configure this object.

**Exceptions:**

*NumberFormatException* if a property that is numeric cannot be converted

*IllegalArgumentException* if a property can't be converted to the correct type.

**6.665.3.3** `virtual double activemq::core::RedeliveryPolicy::getBackOffMultiplier () const` [pure virtual]

**Returns:**

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1678).

**6.665.3.4** `virtual short activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent () const` [pure virtual]

**Returns:**

the currently set Collision Avoidance percentage.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1678).

**6.665.3.5** `virtual long long activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay () const` [pure virtual]

Gets the initial time that redelivery of messages is delayed.

**Returns:**

the time in milliseconds that redelivery is delayed initially.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1678).

**6.665.3.6** `virtual int activemq::core::RedeliveryPolicy::getMaximumRedeliveries () const` [pure virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

**Returns:**

maximum allowed redeliveries for a message.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1679).

**6.665.3.7** `virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay (long long previousDelay)` [pure virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

**Parameters:**

*previousDelay* The last delay that was used between message redeliveries.

**Returns:**

the new delay to use before attempting another redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1679).

**6.665.3.8** `virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance () const` [pure virtual]

**Returns:**

whether or not collision avoidance is enabled for this Policy.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1679).

**6.665.3.9** `virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff () const` [pure virtual]

**Returns:**

whether or not the exponential back off option is enabled.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1679).

**6.665.3.10** `virtual void activemq::core::RedeliveryPolicy::setBackOffMultiplier (double value)` [pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

**Parameters:**

*value* The new value for the back-off multiplier.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1679).

**6.665.3.11** `virtual void activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent (short value)` [pure virtual]

**Parameters:**

*value* The collision avoidance percentage setting.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1680).

**6.665.3.12** `virtual void activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay (long long value)` [pure virtual]

Sets the initial time that redelivery will be delayed.

**Parameters:**

*value* Time in Milliseconds to wait before starting redelivery.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1680).

**6.665.3.13** virtual void activemq::core::RedeliveryPolicy::setMaximumRedeliveries  
(int *maximumRedeliveries*) [pure virtual]

Sets the Maximum allowable redeliveries for a Message.

**Parameters:**

*maximumRedeliveries* The maximum number of times that a message will be redelivered.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1680).

**6.665.3.14** virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance  
(bool *value*) [pure virtual]**Parameters:**

*value* Enable or Disable collision avoidance for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1680).

**6.665.3.15** virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff  
(bool *value*) [pure virtual]**Parameters:**

*value* Enable or Disable the exponential back off multiplier option.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1681).

**6.665.4** Field Documentation**6.665.4.1** const long long activemq::core::RedeliveryPolicy::NO\_MAXIMUM\_REDELIVERIES [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**RedeliveryPolicy.h**

## 6.666 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 2377) with extended capabilities.

#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h> Inheritance diagram for decaf::util::concurrent::locks::ReentrantLock:

### Public Member Functions

- **ReentrantLock** ()
- virtual **~ReentrantLock** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is not held by another thread at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Attempts to release this lock.*
- virtual **Condition** \* **newCondition** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a **Condition** (p. 1249) instance for use with this **Lock** (p. 2377) instance.*
- int **getHoldCount** () const  
*Queries the number of holds on this lock by the current thread.*
- bool **isHeldByCurrentThread** () const  
*Queries if this lock is held by the current thread.*
- bool **isLocked** () const  
*Queries if this lock is held by any thread.*
- bool **isFair** () const  
*Returns true if this lock has fairness set true.*
- std::string **toString** () const  
*Returns a string identifying this lock, as well as its lock state.*

## 6.666.1 Detailed Description

A reentrant mutual exclusion **Lock** (p.2377) with extended capabilities. A **ReentrantLock** (p.3182) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread()** (p.3184), and **getHoldCount()** (p.3183).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, **locks** (p.174) favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair **locks** (p.174) accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain **locks** (p.174) and guarantee lack of starvation. Note however, that fairness of **locks** (p.174) does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:
ReentrantLock (p.3182) lock; // ...

public:

void m() { lock.lock(); // block until condition holds

try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p.2377) interface, this class defines methods **isLocked** and **getLockQueueLength**, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since:

1.0

## 6.666.2 Constructor & Destructor Documentation

**6.666.2.1** decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ()

**6.666.2.2** virtual decaf::util::concurrent::locks::ReentrantLock::~ReentrantLock ()  
[virtual]

## 6.666.3 Member Function Documentation

**6.666.3.1** int decaf::util::concurrent::locks::ReentrantLock::getHoldCount () const

Queries the number of holds on this lock by the current thread. A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of **code** (p.1183) should not be entered with the lock already held then we can assert that fact:

```

class X { private:
ReentrantLock (p. 3182) lock; // ...
public:
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body } catch(...)
{ lock.unlock(); } } }

```

**Returns:**

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

**6.666.3.2 bool decaf::util::concurrent::locks::ReentrantLock::isFair () const**

Returns true if this lock has fairness set true.

**Returns:**

true if this lock has fairness set true

**6.666.3.3 bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread () const**

Queries if this lock is held by the current thread. This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```

class X { private: ReentrantLock (p. 3182) lock = new ReentrantLock() (p. 3183); // ...
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }

```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```

class X { private: ReentrantLock (p. 3182) lock = new ReentrantLock() (p. 3183); // ...
public: void m() { assert !lock.isHeldByCurrentThread() (p. 3184); lock.lock(); try { // ...
method body } finally { lock.unlock(); } } }

```

**Returns:**

true if current thread holds this lock and false otherwise

**6.666.3.4 bool decaf::util::concurrent::locks::ReentrantLock::isLocked () const**

Queries if this lock is held by any thread. This method is designed for use in monitoring of the system state, not for synchronization control.

**Returns:**

true if any thread holds this lock and false otherwise



**6.666.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock () throw ( decaf::lang::exceptions::RuntimeException ) [virtual]**

Acquires the lock. Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

**Exceptions:**

*RuntimeException* if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2378).

**6.666.3.6 virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [virtual]**

Acquires the lock unless the current thread is interrupted. Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then *InterruptedException* is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

**Exceptions:**

*RuntimeException* if an error occurs while acquiring the lock.

*InterruptedException* if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implements **decaf::util::concurrent::locks::Lock** (p. 2379).

**6.666.3.7** `virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition()` `throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Returns a **Condition** (p. 1249) instance for use with this **Lock** (p. 2377) instance. The returned **Condition** (p. 1249) instance supports the same usages as do the **Mutex** (p. 2782) Class' methods (wait, notify, and notifyAll).

\* If this lock is not held when any of the **Condition** (p. 1249) waiting or signalling methods are called, then an `IllegalMonitorStateException` is thrown. \* When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. \* If a thread is interrupted while waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. \* Waiting threads are signaled in FIFO order. \* The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair **locks** (p. 174) favors those threads that have been waiting the longest.

#### Exceptions:

***RuntimeException*** if an error occurs while creating the **Condition** (p. 1249).

***UnsupportedOperationException*** if this **Lock** (p. 2377) implementation does not support conditions

Implements `decaf::util::concurrent::locks::Lock` (p. 2379).

**6.666.3.8** `std::string decaf::util::concurrent::locks::ReentrantLock::toString ()` `const`

Returns a string identifying this lock, as well as its lock state. The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

#### Returns:

a string identifying this lock, as well as its lock state

**6.666.3.9** `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock(long long time, const TimeUnit & unit)` `throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )` [virtual]

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted. Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 3187) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread;  
or \* The specified waiting time elapses

If the lock is acquired then the value `true` is returned and the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value `false` is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

#### Parameters:

*time* the maximum time to wait for the lock

*unit* the time unit of the time argument

#### Returns:

`true` if the lock was acquired and `false` if the waiting time elapsed before the lock was acquired

#### Exceptions:

*RuntimeException* if an error occurs while acquiring the lock.

*InterruptedException* if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements `decaf::util::concurrent::locks::Lock` (p. 2380).

#### 6.666.3.10 `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ()` `throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Acquires the lock only if it is not held by another thread at the time of invocation. Acquires the lock if it is not held by another thread and returns immediately with the value `true`, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to `tryLock()` (p. 3187) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use `tryLock(0, TimeUnit.SECONDS)` (p. 3816) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns `true`.

If the lock is held by another thread then this method will return immediately with the value `false`.

#### Returns:

`true` if the lock was acquired and `false` otherwise

#### Exceptions:

*RuntimeException* if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2381).

**6.666.3.11**    **virtual void decaf::util::concurrent::locks::ReentrantLock::unlock**  
                  **() throw ( decaf::lang::exceptions::RuntimeException,**  
                  **decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]**

Attempts to release this lock. If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then `IllegalMonitorStateException` is thrown.

**Exceptions:**

***RuntimeException*** if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2381).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReentrantLock.h`

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference

#include <src/main/decaf/util/concurrent/RejectedExecutionException.h> Inheritance diagram for decaf::util::concurrent::RejectedExecutionException:

### Public Member Functions

- **RejectedExecutionException** () throw ()  
*Default Constructor.*
- **RejectedExecutionException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()  
*Copy Constructor.*
- **RejectedExecutionException** (const std::exception \*cause) throw ()  
*Constructor.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RejectedExecutionException** \* clone () const  
*Clones this exception.*
- virtual ~**RejectedExecutionException** () throw ()

### 6.667.1 Constructor & Destructor Documentation

#### 6.667.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException () throw () [inline]

Default Constructor.

#### 6.667.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

**6.667.1.3** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & ex) throw ()` [inline]

Copy Constructor.

**Parameters:**

*ex* - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

**6.667.1.4** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const std::exception * cause) throw ()` [inline]

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.667.1.5** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*msg* - The message to report

*...* - list of primitives that are formatted into the message

**6.667.1.6** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* - The file name where exception occurs

*lineNumber* - The line number where the exception occurred.

*cause* - The exception that was the cause for this one to be thrown.

*msg* - The message to report

*...* - list of primitives that are formatted into the message

**6.667.1.7** virtual  
decaf::util::concurrent::RejectedExecutionException::~RejectedExecutionException  
( ) throw () [inline, virtual]

## 6.667.2 Member Function Documentation

**6.667.2.1** virtual RejectedExecutionException\* de-  
caf::util::concurrent::RejectedExecutionException::clone ()  
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1834).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionException.h**

## 6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>

### Public Member Functions

- virtual **~RejectedExecutionHandler** ()
- virtual void **rejectedExecution** (Runnable \*r, ThreadPoolExecutor \*executer)=0 throw (RejectedExecutionException )

*Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task.*

### 6.668.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

Since:

1.0

### 6.668.2 Constructor & Destructor Documentation

- 6.668.2.1** virtual  
decaf::util::concurrent::RejectedExecutionHandler::~~RejectedExecutionHandler  
( ) [inline, virtual]

### 6.668.3 Member Function Documentation

- 6.668.3.1** virtual void de-  
caf::util::concurrent::RejectedExecutionHandler::rejectedExecution  
(Runnable \* r, ThreadPoolExecutor \* *executer*) throw ( RejectedExecutionException ) [pure virtual]

Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task. This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1869).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 3189), which will be propagated to the caller of **execute** (p. ??).

Parameters:

*r* The pointer to the runnable task requested to be executed.

*executer* The pointer to the executor attempting to execute this task.

Exceptions:

*RejectedExecutionException* (p. 3189) if there is no remedy.



The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionHandler.h`

## 6.669 activemq::commands::RemoveInfo Class Reference

`#include <src/main/activemq/commands/RemoveInfo.h>` Inheritance diagram for `activemq::commands::RemoveInfo`:

### Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **RemoveInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)
- virtual long long **getLastDeliveredSequenceId** () const
- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REMOVEINFO** = 12

### Protected Attributes

- **Pointer**< **DataStructure** > **objectId**
- long long **lastDeliveredSequenceId**

## 6.669.1 Constructor & Destructor Documentation

**6.669.1.1** `activemq::commands::RemoveInfo::RemoveInfo ()`

**6.669.1.2** `virtual activemq::commands::RemoveInfo::~~RemoveInfo () [virtual]`

## 6.669.2 Member Function Documentation

**6.669.2.1** `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.669.2.2** `virtual void activemq::commands::RemoveInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.669.2.3** `virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.669.2.4** `virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

**6.669.2.5** `virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceId () const`  
[virtual]

**6.669.2.6** `virtual Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ()`  
[virtual]

**6.669.2.7** `virtual const Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId () const`  
[virtual]

**6.669.2.8** `virtual bool activemq::commands::RemoveInfo::isRemoveInfo () const`  
[inline, virtual]

**Returns:**

an answer of true to the `isRemoveInfo()` (p. 3196) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 762).

**6.669.2.9** `virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceId (long long lastDeliveredSequenceId)` [virtual]

**6.669.2.10** `virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataStructure > & objectId)` [virtual]

**6.669.2.11** `virtual std::string activemq::commands::RemoveInfo::toString () const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

**6.669.2.12** `virtual Pointer<Command> activemq::commands::RemoveInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1198).

### 6.669.3 Field Documentation

- 6.669.3.1 `const unsigned char activemq::commands::RemoveInfo::ID_REMOVEINFO = 12` [static]
- 6.669.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId` [protected]
- 6.669.3.3 `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

## 6.670 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3198).

#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.670.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3198). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.670.2 Constructor & Destructor Documentation

**6.670.2.1** `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller()` `[inline]`

**6.670.2.2** `virtual activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` `[inline, virtual]`

## 6.670.3 Member Function Documentation

**6.670.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.670.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.670.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.670.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.670.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).



**6.670.3.6** virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.670.3.7** virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**

## 6.671 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3202).

#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.671.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3202). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.671.2 Constructor & Destructor Documentation

**6.671.2.1** `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

**6.671.2.2** `virtual activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

## 6.671.3 Member Function Documentation

**6.671.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.671.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.671.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.671.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.671.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.671.3.6** virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.671.3.7** virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**RemoveInfoMarshaller.h**

## 6.672 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3206).

#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.672.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3206). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.672.2 Constructor & Destructor Documentation

**6.672.2.1** `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

**6.672.2.2** `virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

## 6.672.3 Member Function Documentation

**6.672.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.672.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.672.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.672.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.672.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).



**6.672.3.6** virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.672.3.7** virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

## 6.673 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3210).

#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.673.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3210). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.673.2 Constructor & Destructor Documentation

**6.673.2.1** `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller()` `[inline]`

**6.673.2.2** `virtual activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` `[inline, virtual]`

## 6.673.3 Member Function Documentation

**6.673.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.673.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.673.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.673.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.673.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.673.3.6** virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.673.3.7** virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**

## 6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3214).

#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.674.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3214). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.674.2 Constructor & Destructor Documentation

**6.674.2.1** `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller()` `[inline]`

**6.674.2.2** `virtual activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` `[inline, virtual]`

## 6.674.3 Member Function Documentation

**6.674.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.674.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.674.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.674.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.674.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).



**6.674.3.6** virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.674.3.7** virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**RemoveInfoMarshaller.h**

## 6.675 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3218).

#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.675.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3218). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.675.2 Constructor & Destructor Documentation

**6.675.2.1** `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller()` `[inline]`

**6.675.2.2** `virtual activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` `[inline, virtual]`

## 6.675.3 Member Function Documentation

**6.675.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.675.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.675.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.675.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.675.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.675.3.6** virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p.776).

**6.675.3.7** virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p.777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveInfoMarshaller.h**

## 6.676 activemq::commands::RemoveSubscriptionInfo Class Reference

#include <src/main/activemq/commands/RemoveSubscriptionInfo.h> Inheritance diagram for activemq::commands::RemoveSubscriptionInfo:

### Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **RemoveSubscriptionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REMOVESUBSCRIPTIONINFO** = 9

## Protected Attributes

- `Pointer< ConnectionId > connectionId`
- `std::string subscriptionName`
- `std::string clientId`

## 6.676.1 Constructor & Destructor Documentation

**6.676.1.1** `activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo()`

**6.676.1.2** `virtual`  
`activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo()` `[virtual]`

## 6.676.2 Member Function Documentation

**6.676.2.1** `virtual RemoveSubscriptionInfo* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure()` `const` `[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.676.2.2** `virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure(const DataStructure * src)` `[virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.676.2.3** `virtual bool activemq::commands::RemoveSubscriptionInfo::equals(const DataStructure * value)` `const` `[virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.676.2.4 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]`
- 6.676.2.5 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]`
- 6.676.2.6 `virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]`
- 6.676.2.7 `virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]`
- 6.676.2.8 `virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.676.2.9 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () [virtual]`
- 6.676.2.10 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () const [virtual]`
- 6.676.2.11 `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const [inline, virtual]`

#### Returns:

an answer of true to the **isRemoveSubscriptionInfo()** (p. 3224) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 762).



**6.676.2.12** virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & *clientId*) [virtual]

**6.676.2.13** virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

**6.676.2.14** virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName (const std::string & *subscriptionName*) [virtual]

**6.676.2.15** virtual std::string activemq::commands::RemoveSubscriptionInfo::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.676.2.16** virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor \* *visitor*) throw ( exceptions::ActiveMQException ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

### 6.676.3 Field Documentation

- 6.676.3.1** `std::string activemq::commands::RemoveSubscriptionInfo::clientId`  
[protected]
- 6.676.3.2** `Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId`  
[protected]
- 6.676.3.3** `const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVE_SUBSCRIPTION_INFO = 9` [static]
- 6.676.3.4** `std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveSubscriptionInfo.h`

6.677

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

3229

6.677 ~~activemq::wireformat::openwire::marshal::v1::RemoveSubscription~~

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3227).

#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller:

## Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.677.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3227). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.677.2 Constructor & Destructor Documentation

**6.677.2.1** `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

**6.677.2.2** `virtual activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

## 6.677.3 Member Function Documentation

**6.677.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.677.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.677.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.677

**activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

**Class Reference**

3231

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 780).

**6.677.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 781).

**6.677.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

### Returns:

int value indicating the size of the marshaled object.

### Exceptions:

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 782).

**6.677.3.6** virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.677.3.7** virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**

6.678

activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

3233

6.678 — activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3231).

#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller:

## Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
  - virtual **~RemoveSubscriptionInfoMarshaller** ()
  - virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.678.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3231). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.678.2 Constructor & Destructor Documentation

**6.678.2.1** `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

**6.678.2.2** `virtual activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

## 6.678.3 Member Function Documentation

**6.678.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.678.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.678.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



6.678

activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

3235

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

6.678.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

6.678.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.678.3.6** virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.678.3.7** virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveSubscriptionInfoMarshaller.h**

6.679

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

3237

6.679 — activemq::wireformat::openwire::marshal::v2::RemoveSubscription

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3235).

#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.679.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3235). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.679.2 Constructor & Destructor Documentation

**6.679.2.1** `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfo()` [inline]

**6.679.2.2** `virtual activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfo()` [inline, virtual]

## 6.679.3 Member Function Documentation

**6.679.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.679.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.679.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

6.679

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

3239

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.679.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.679.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.679.3.6** virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.679.3.7** virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**

6.680

activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

Class Reference

6.680 — activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3239).

#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
  - virtual ~**RemoveSubscriptionInfoMarshaller** ()
  - virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.680.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3239). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.680.2 Constructor & Destructor Documentation

**6.680.2.1** `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

**6.680.2.2** `virtual activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

## 6.680.3 Member Function Documentation

**6.680.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.680.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.680.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



6.680

activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

Class Reference

3243

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

6.680.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

6.680.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.680.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.680.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h`

6.681

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

Class Reference

6.681 ~~activemq::wireformat::openwire::marshal::v5::RemoveSubscription~~ <sup>3245</sup>

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3243).

#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.681.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3243). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.681.2 Constructor & Destructor Documentation

**6.681.2.1** `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfo()` [inline]

**6.681.2.2** `virtual activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfo()` [inline, virtual]

## 6.681.3 Member Function Documentation

**6.681.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.681.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.681.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

6.681

**activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

**Class Reference**

3247

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 787).

**6.681.3.4** virtual void ac-

tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 788).

**6.681.3.5** virtual int ac-

tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 789).

**6.681.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.681.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h`

6.682

activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

3249

6.682 — activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

## Class Reference

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3247).

#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h>In diagram for activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.682.1 Detailed Description

Marshaling **code** (p.1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3247). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.682.2 Constructor & Destructor Documentation

**6.682.2.1** `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfo()` [inline]

**6.682.2.2** `virtual activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfo()` [inline, virtual]

## 6.682.3 Member Function Documentation

**6.682.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.682.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.682.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



6.682

**activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

**Class Reference**

3251

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 773).

**6.682.3.4** virtual void ac-

```
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 774).

**6.682.3.5** virtual int ac-

```
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 775).

**6.682.3.6** virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.682.3.7** virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveSubscriptionInfoMarshaller.h**

## 6.683 activemq::commands::ReplayCommand Class Reference

#include <src/main/activemq/commands/ReplayCommand.h> Inheritance diagram for activemq::commands::ReplayCommand:

### Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ReplayCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID \_REPLAYCOMMAND** = 65

### Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

## 6.683.1 Constructor & Destructor Documentation

**6.683.1.1** `activemq::commands::ReplayCommand::ReplayCommand ()`

**6.683.1.2** `virtual activemq::commands::ReplayCommand::~~ReplayCommand ()`  
[virtual]

## 6.683.2 Member Function Documentation

**6.683.2.1** `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.683.2.2** `virtual void activemq::commands::ReplayCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.683.2.3** `virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.683.2.4** `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p.1662).

- 6.683.2.5**    `virtual int activemq::commands::ReplayCommand::getFirstNakNumber  
() const [virtual]`
- 6.683.2.6**    `virtual int activemq::commands::ReplayCommand::getLastNakNumber  
() const [virtual]`
- 6.683.2.7**    `virtual void activemq::commands::ReplayCommand::setFirstNakNumber  
(int firstNakNumber) [virtual]`
- 6.683.2.8**    `virtual void activemq::commands::ReplayCommand::setLastNakNumber  
(int lastNakNumber) [virtual]`
- 6.683.2.9**    `virtual std::string activemq::commands::ReplayCommand::toString ()  
const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.683.2.10**    `virtual Pointer<Command> ac-  
tivemq::commands::ReplayCommand::visit  
(activemq::state::CommandVisitor * visitor) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

## 6.683.3 Field Documentation

- 6.683.3.1**    `int activemq::commands::ReplayCommand::firstNakNumber [protected]`
- 6.683.3.2**    `const unsigned char activemq::commands::ReplayCommand::ID _ -  
REPLAYCOMMAND = 65 [static]`
- 6.683.3.3**    `int activemq::commands::ReplayCommand::lastNakNumber [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ReplayCommand.h`

## 6.684 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3254).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.684.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3254).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.684.2 Constructor & Destructor Documentation

**6.684.2.1** `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

**6.684.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

## 6.684.3 Member Function Documentation

**6.684.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.684.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.684.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.684.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.684.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).



**6.684.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.684.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h

## 6.685 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3258).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.685.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3258).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.685.2 Constructor & Destructor Documentation

**6.685.2.1** `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller()` `[inline]`

**6.685.2.2** `virtual`  
`activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` `[inline, virtual]`

## 6.685.3 Member Function Documentation

**6.685.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.685.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.685.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.685.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.685.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.685.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.685.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h

## 6.686 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3262).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.686.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3262).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.686.2 Constructor & Destructor Documentation

**6.686.2.1** `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

**6.686.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

## 6.686.3 Member Function Documentation

**6.686.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.686.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.686.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.686.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.686.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).



**6.686.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.686.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h

## 6.687 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3266).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.687.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3266).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.687.2 Constructor & Destructor Documentation

**6.687.2.1** `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

**6.687.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

## 6.687.3 Member Function Documentation

**6.687.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.687.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.687.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.687.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.687.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.687.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.687.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h

## 6.688 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3270).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.688.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3270).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.688.2 Constructor & Destructor Documentation

**6.688.2.1** `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

**6.688.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

## 6.688.3 Member Function Documentation

**6.688.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.688.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.688.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.688.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.688.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).



**6.688.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.688.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h

## 6.689 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3274).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.689.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3274).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.689.2 Constructor & Destructor Documentation

**6.689.2.1** `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

**6.689.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

## 6.689.3 Member Function Documentation

**6.689.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.689.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.689.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.689.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.689.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.689.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.689.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h

## 6.690 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveProducerExecutor:

### Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** \**action*, **CmsTemplate** \**parent*, const std::string &*destinationName*)
- virtual ~**ResolveProducerExecutor** ()
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \**session*) throw ( cms::CMSException )

### Protected Member Functions

- **ResolveProducerExecutor** & **operator=** (const **ResolveProducerExecutor** &)

### 6.690.1 Constructor & Destructor Documentation

- 6.690.1.1** **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor** (**ProducerCallback** \* *action*, **CmsTemplate** \* *parent*, const std::string & *destinationName*) [inline]
- 6.690.1.2** virtual **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor** () [inline, virtual]

### 6.690.2 Member Function Documentation

- 6.690.2.1** virtual **cms::Destination\*** **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination** (**cms::Session** \* *session*) throw ( cms::CMSException ) [virtual]
- 6.690.2.2** **ResolveProducerExecutor&** **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::operator=** (const **ResolveProducerExecutor** &) [inline, protected]

Reimplemented from **activemq::cmsutil::CmsTemplate::ProducerExecutor** (p. 3066).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.691 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor:

### Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** \*parent, const std::string &selector, bool noLocal, const std::string &destinationName)
- virtual ~**ResolveReceiveExecutor** ()
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session) throw ( cms::CMSException )

### Protected Member Functions

- **ResolveReceiveExecutor** & **operator=** (const **ResolveReceiveExecutor** &)

### 6.691.1 Constructor & Destructor Documentation

- 6.691.1.1** **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor** (**CmsTemplate** \* *parent*, const std::string & *selector*, bool *noLocal*, const std::string & *destinationName*) [inline]
- 6.691.1.2** virtual **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~~ResolveReceiveExecutor** () [inline, virtual]

### 6.691.2 Member Function Documentation

- 6.691.2.1** virtual **cms::Destination\*** **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination** (**cms::Session** \* *session*) throw ( cms::CMSException ) [virtual]
- 6.691.2.2** **ResolveReceiveExecutor&** **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::operator=** (const **ResolveReceiveExecutor** &) [inline, protected]

Reimplemented from **activemq::cmsutil::CmsTemplate::ReceiveExecutor** (p. 3175).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.692 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

`#include <src/main/decaf/internal/util/Resource.h>`Inheritance diagram for decaf::internal::util::Resource:

### Public Member Functions

- virtual `~Resource()`

#### 6.692.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

**Since:**

1.0

#### 6.692.2 Constructor & Destructor Documentation

##### 6.692.2.1 virtual `decaf::internal::util::Resource::~~Resource()` [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/Resource.h`



## 6.693 decaf::internal::util::ResourceLifecycleManager Class Reference

```
#include <src/main/decaf/internal/util/ResourceLifecycleManager.h>
```

### Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** \*value)

### Protected Member Functions

- virtual void **destroyResources** ()

### 6.693.1 Detailed Description

Since:

1.0

### 6.693.2 Constructor & Destructor Documentation

**6.693.2.1** decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager ()

**6.693.2.2** virtual decaf::internal::util::ResourceLifecycleManager::~~ResourceLifecycleManager () [virtual]

### 6.693.3 Member Function Documentation

**6.693.3.1** virtual void decaf::internal::util::ResourceLifecycleManager::addResource (**Resource** \* *value*) [virtual]

**6.693.3.2** virtual void decaf::internal::util::ResourceLifecycleManager::destroyResources () [protected, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ResourceLifecycleManager.h**

## 6.694 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```

### Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()  
*Destructor - calls **destroy**.*
- void **addConnection** (cms::Connection \*connection) throw ( cms::CMSEException )  
*Adds a connection so that its life will be managed by this object.*
- void **addSession** (cms::Session \*session) throw ( cms::CMSEException )  
*Adds a session so that its life will be managed by this object.*
- void **addDestination** (cms::Destination \*dest) throw ( cms::CMSEException )  
*Adds a destination so that its life will be managed by this object.*
- void **addMessageProducer** (cms::MessageProducer \*producer) throw ( cms::CMSEException )  
*Adds a message producer so that its life will be managed by this object.*
- void **addMessageConsumer** (cms::MessageConsumer \*consumer) throw ( cms::CMSEException )  
*Adds a message consumer so that its life will be managed by this object.*
- void **destroy** () throw ( cms::CMSEException )  
*Closes and destroys the contained CMS resources.*
- void **releaseAll** ()  
*Releases all of the contained resources so that this object will no longer control their lifetimes.*

### Protected Member Functions

- **ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & operator= (const **ResourceLifecycleManager** &)

#### 6.694.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to **destroy** will close and destroy all of the contained resources in the appropriate manner.

## 6.694.2 Constructor & Destructor Documentation

**6.694.2.1** `activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager (const ResourceLifecycleManager &) [inline, protected]`

**6.694.2.2** `activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager ()`

**6.694.2.3** `virtual  
activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager  
() [virtual]`

Destructor - calls destroy.

## 6.694.3 Member Function Documentation

**6.694.3.1** `void activemq::cmsutil::ResourceLifecycleManager::addConnection (cms::Connection * connection) throw ( cms::CMSException )`

Adds a connection so that its life will be managed by this object.

**Parameters:**

*connection* the object to be managed

**6.694.3.2** `void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * dest) throw ( cms::CMSException )`

Adds a destination so that its life will be managed by this object.

**Parameters:**

*dest* the object to be managed

**6.694.3.3** `void ac-  
tivemq::cmsutil::ResourceLifecycleManager::addMessageConsumer  
(cms::MessageConsumer * consumer) throw ( cms::CMSException )`

Adds a message consumer so that its life will be managed by this object.

**Parameters:**

*consumer* the object to be managed

**6.694.3.4** `void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * producer) throw ( cms::CMSException )`

Adds a message producer so that its life will be managed by this object.

**Parameters:**

*producer* the object to be managed

**6.694.3.5** `void activemq::cmsutil::ResourceLifecycleManager::addSession  
(cms::Session * session) throw ( cms::CMSException )`

Adds a session so that its life will be managed by this object.

**Parameters:**

*session* the object to be managed

**6.694.3.6** `void activemq::cmsutil::ResourceLifecycleManager::destroy () throw ( cms::CMSException )`

Closes and destroys the contained CMS resources.

**Exceptions:**

*cms::CMSException* (p. 1160) thrown if an error occurs.

**6.694.3.7** `ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator=  
(const ResourceLifecycleManager &) [inline, protected]`

**6.694.3.8** `void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()`

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ResourceLifecycleManager.h`

## 6.695 activemq::commands::Response Class Reference

#include <src/main/activemq/commands/Response.h> Inheritance diagram for activemq::commands::Response:

### Public Member Functions

- **Response** ()
- virtual **~Response** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Response \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int correlationId)
- virtual bool **isResponse** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_RESPONSE** = 30

### Protected Attributes

- int **correlationId**

## 6.695.1 Constructor & Destructor Documentation

**6.695.1.1** `activemq::commands::Response::Response ()`

**6.695.1.2** `virtual activemq::commands::Response::~~Response () [virtual]`

## 6.695.2 Member Function Documentation

**6.695.2.1** `virtual Response* activemq::commands::Response::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1529), `activemq::commands::DataResponse` (p. 1584), `activemq::commands::ExceptionResponse` (p. 1840), and `activemq::commands::IntegerResponse` (p. 2092).

**6.695.2.2** `virtual void activemq::commands::Response::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1529), `activemq::commands::DataResponse` (p. 1584), `activemq::commands::ExceptionResponse` (p. 1840), and `activemq::commands::IntegerResponse` (p. 2092).

**6.695.2.3** `virtual bool activemq::commands::Response::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1529), **activemq::commands::DataResponse** (p. 1584), **activemq::commands::ExceptionResponse** (p. 1840), and **activemq::commands::IntegerResponse** (p. 2092).

**6.695.2.4** `virtual int activemq::commands::Response::getCorrelationId () const`  
[virtual]

**6.695.2.5** `virtual unsigned char activemq::commands::Response::getDataStructureType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1530), **activemq::commands::DataResponse** (p. 1585), **activemq::commands::ExceptionResponse** (p. 1840), and **activemq::commands::IntegerResponse** (p. 2092).

**6.695.2.6** `virtual bool activemq::commands::Response::isResponse () const`  
[inline, virtual]

#### Returns:

an answer of true to the **isResponse()** (p. 3287) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 762).

**6.695.2.7** `virtual void activemq::commands::Response::setCorrelationId (int correlationId)` [virtual]

**6.695.2.8** `virtual std::string activemq::commands::Response::toString () const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1530), **activemq::commands::DataResponse** (p. 1585), **activemq::commands::ExceptionResponse** (p. 1841), and **activemq::commands::IntegerResponse** (p. 2093).

**6.695.2.9** `virtual Pointer<Command> activemq::commands::Response::visit  
(activemq::state::CommandVisitor * visitor) throw (  
exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1198).

### 6.695.3 Field Documentation

**6.695.3.1** `int activemq::commands::Response::correlationId` [protected]

**6.695.3.2** `const unsigned char activemq::commands::Response::ID_RESPONSE =  
30` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Response.h`



## 6.696 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

#include <src/main/activemq/transport/mock/ResponseBuilder.h> Inheritance diagram for activemq::transport::mock::ResponseBuilder:

### Public Member Functions

- virtual **~ResponseBuilder** ()
- virtual **Pointer< Response > buildResponse** (const **Pointer< Command > &command**)=0  
*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*
- virtual void **buildIncomingCommands** (const **Pointer< Command > &command**, **decaf::util::StlQueue< Pointer< Command > > &queue**)=0  
*When called the **ResponseBuilder** (p. 3289) must construct all the Responses or Asynchronous **commands** (p. 87) that would be sent to this client by the Broker upon receipt of the passed command.*

### 6.696.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

### 6.696.2 Constructor & Destructor Documentation

**6.696.2.1** virtual **activemq::transport::mock::ResponseBuilder::~~ResponseBuilder** () [inline, virtual]

### 6.696.3 Member Function Documentation

**6.696.3.1** virtual void **activemq::transport::mock::ResponseBuilder::buildIncomingCommands** (const **Pointer< Command > &command**, **decaf::util::StlQueue< Pointer< Command > > &queue**) [pure virtual]

When called the **ResponseBuilder** (p. 3289) must construct all the Responses or Asynchronous **commands** (p. 87) that would be sent to this client by the Broker upon receipt of the passed command.

#### Parameters:

- command* - The Command being sent to the Broker.
- queue* - Queue of Command sent back from the broker.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2905).

**6.696.3.2** `virtual Pointer<Response> activemq::transport::mock::ResponseBuilder::buildResponse(const Pointer< Command > & command)` [pure virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

**Parameters:**

*command* - The command to build a response for

**Returns:**

A Response object pointer, or NULL if no response.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2906).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/ResponseBuilder.h`

## 6.697 activemq::transport::correlator::ResponseCorrelator Class Reference

This type of **transport** (p.97) filter is responsible for correlating asynchronous responses with requests.

#include <src/main/activemq/transport/correlator/ResponseCorrelator.h> Inheritance diagram for activemq::transport::correlator::ResponseCorrelator:

### Public Member Functions

- **ResponseCorrelator** (const **Pointer**< **Transport** > &next)  
*Constructor.*
- virtual ~**ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this **transport** (p.97) object and creates the thread for polling on the input stream for **commands** (p.87).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onTransportException** (**Transport** \*source, const decaf::lang::Exception &ex)  
*Event handler for an exception from a command **transport** (p.97).*

### 6.697.1 Detailed Description

This type of **transport** (p.97) filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the **transport** (p.97) that it

## 6.697.2 Constructor & Destructor Documentation

### 6.697.2.1 `activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & next)`

Constructor.

#### Parameters:

*next* the next **transport** (p. 97) in the chain

### 6.697.2.2 `virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator ()` [virtual]

## 6.697.3 Member Function Documentation

### 6.697.3.1 `virtual void activemq::transport::correlator::ResponseCorrelator::close () throw ( decaf::io::IOException )` [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions:

*IOException* if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3893).

### 6.697.3.2 `virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & command)` [virtual]

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

#### Parameters:

*command* the received from the nested **transport** (p. 97).

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

### 6.697.3.3 `virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Sends a one-way command. Does not wait for any response from the broker.

#### Parameters:

*command* the command to be sent.

**Exceptions:**

***IOException*** if an exception occurs during writing of the command.

***UnsupportedOperationException*** if this method is not implemented by this **transport** (p. 97).

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

**6.697.3.4** virtual void **activemq::transport::correlator::ResponseCorrelator::onTransportException** (**Transport** \* *source*, const **decaf::lang::Exception** & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 97).

**Parameters:**

*source* The source of the exception

*ex* The exception.

**6.697.3.5** virtual **Pointer<Response>** **activemq::transport::correlator::ResponseCorrelator::request** (const **Pointer< Command >** & *command*, unsigned int *timeout*) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** ) [virtual]

Sends the given request to the server and waits for the response.

**Parameters:**

*command* The request to send.

*timeout* The time to wait for a response.

**Returns:**

the response from the server.

**Exceptions:**

***IOException*** if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3896).

**6.697.3.6** virtual **Pointer<Response>** **activemq::transport::correlator::ResponseCorrelator::request** (const **Pointer< Command >** & *command*) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** ) [virtual]

Sends the given request to the server and waits for the response.

**Parameters:**

*command* The request to send.

**Returns:**

the response from the server.

**Exceptions:**

***IOException*** if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3897).

**6.697.3.7** **virtual void activemq::transport::correlator::ResponseCorrelator::start ()**  
**throw ( decaf::io::IOException )** [virtual]

Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions:**

***IOException*** if an error occurs or if this **transport** (p. 97) has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3898).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

## 6.698 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3295).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.698.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3295). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.698.2 Constructor & Destructor Documentation

**6.698.2.1** `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.698.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.698.3 Member Function Documentation

**6.698.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1540), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1603), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1859), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2107).

**6.698.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1540), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1603), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1859), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2107).

**6.698.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.



**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 773).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1540), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1603), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1859), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2107).

**6.698.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 774).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1541), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1604), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1860), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2108).

**6.698.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 775).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1541), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1604), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1860), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2108).

**6.698.3.6** virtual void **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1542), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1605), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1861), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2109).

**6.698.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1542), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1605), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1861), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2109).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ResponseMarshaller.h**

## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3300).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.699.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3300). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.699.2 Constructor & Destructor Documentation

**6.699.2.1** `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.699.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.699.3 Member Function Documentation

**6.699.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1532), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1847), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2099).

**6.699.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1532), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1847), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2099).

**6.699.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 801).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1595), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1847), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2099).

**6.699.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 802).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1533), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1596), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1848), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2100).

**6.699.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 803).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1533), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1596), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1848), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2100).

**6.699.3.6 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2**  
**(OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut,**  
**utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1534), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1597), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1849), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2101).

```
6.699.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn,
 utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 805).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1534), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1597), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1849), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2101).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h`



## 6.700 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3305).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.700.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3305). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.700.2 Constructor & Destructor Documentation

**6.700.2.1** `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.700.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.700.3 Member Function Documentation

**6.700.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1548), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1587), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1855), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2115).

**6.700.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1548), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1587), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1855), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2115).

**6.700.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 787).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1548), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1587), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1855), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2115).

**6.700.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 788).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1549), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1588), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1856), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2116).

**6.700.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 789).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1549), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1588), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1856), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2116).

**6.700.3.6 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2**  
**(OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut,**  
**utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1550), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1589), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1857), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2117).

```
6.700.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.

*dataStructure* - Object to be un-marshaled.

*dataIn* - BinaryReader that provides that data.

*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1550), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1589), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1857), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2117).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ResponseMarshaller.h**

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3310).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.701.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3310). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.701.2 Constructor & Destructor Documentation

**6.701.2.1** `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.701.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.701.3 Member Function Documentation

**6.701.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1536), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1851), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2103).

**6.701.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1536), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1851), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2103).

**6.701.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 766).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1536), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1599), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1851), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2103).

**6.701.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 767).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1537), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1600), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1852), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2104).

**6.701.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.



**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 768).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1537), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1600), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1852), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2104).

**6.701.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1538), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1601), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1853), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2105).

**6.701.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1538), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1601), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1853), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2105).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ResponseMarshaller.h**

## 6.702 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3315).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.702.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3315). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.702.2 Constructor & Destructor Documentation

**6.702.2.1** `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.702.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.702.3 Member Function Documentation

**6.702.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1544), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1607), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1863), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2111).

**6.702.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1544), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1607), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1863), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2111).

**6.702.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 780).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1544), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1607), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1863), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2111).

**6.702.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 781).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1545), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1608), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1864), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2112).

**6.702.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 782).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1545), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1608), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1864), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2112).

**6.702.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1546), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1609), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1865), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2113).

**6.702.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1546), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1609), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1865), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2113).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ResponseMarshaller.h**

## 6.703 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3320).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.703.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3320). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.703.2 Constructor & Destructor Documentation

**6.703.2.1** `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::ResponseMarshaller()` [inline]

**6.703.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

## 6.703.3 Member Function Documentation

**6.703.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1552), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1843), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2095).

**6.703.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1552), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1843), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2095).

**6.703.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 794).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1552), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1591), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1843), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2095).

**6.703.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseUnmarshal (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn) throw ( decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 795).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1553), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1592), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1844), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2096).

**6.703.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal1 (OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 796).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1553), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1592), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1844), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2096).

**6.703.3.6 virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal2**  
**(OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut,**  
**utils::BooleanStream \* bs) throw ( decaf::io::IOException ) [virtual]**

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1593), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1845), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2097).

```
6.703.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn,
 utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 798).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1554), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1593), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1845), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2097).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h`

## 6.704 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

#include <src/main/decaf/lang/Runnable.h>Inheritance diagram for decaf::lang::Runnable:

### Public Member Functions

- virtual `~Runnable()`
- virtual void `run()`=0

*Run method - called by the **Thread** (p. 3765) class in the context of the thread.*

### 6.704.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

### 6.704.2 Constructor & Destructor Documentation

**6.704.2.1** virtual decaf::lang::Runnable::~~Runnable() [inline, virtual]

### 6.704.3 Member Function Documentation

**6.704.3.1** virtual void decaf::lang::Runnable::run() [pure virtual]

Run method - called by the **Thread** (p. 3765) class in the context of the thread.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1224), **activemq::threads::DedicatedTaskRunner** (p. 1671), **activemq::transport::inactivity::ReadChecker** (p. 3162), **activemq::transport::inactivity::WriteChecker** (p. 4020), **activemq::transport::IOTransport** (p. 2150), **activemq::transport::mock::InternalCommandListener** (p. 2123), **decaf::lang::Thread** (p. 3771), and **decaf::util::concurrent::PooledThread** (p. 2969).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runnable.h**

## 6.705 decaf::lang::Runtime Class Reference

#include <src/main/decaf/lang/Runtime.h> Inheritance diagram for decaf::lang::Runtime:

### Public Member Functions

- virtual `~Runtime ()`

### Static Public Member Functions

- static `Runtime * getRuntime ()`  
*Gets the single instance of the Decaf **Runtime** (p. 3326) for this Process.*
- static void `initializeRuntime (int argc, char **argv)`  
*Initialize the Decaf Library passing it the args that were passed to the application at startup.*
- static void `initializeRuntime ()`  
*Initialize the Decaf Library.*
- static void `shutdownRuntime ()`  
*Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.*

### 6.705.1 Constructor & Destructor Documentation

**6.705.1.1** virtual `decaf::lang::Runtime::~~Runtime ()` [inline, virtual]

### 6.705.2 Member Function Documentation

**6.705.2.1** static `Runtime* decaf::lang::Runtime::getRuntime ()` [static]

Gets the single instance of the Decaf **Runtime** (p. 3326) for this Process.

#### Returns:

pointer to the single Decaf **Runtime** (p. 3326) instance that exists for this process

**6.705.2.2** static void `decaf::lang::Runtime::initializeRuntime ()` [static]

Initialize the Decaf Library.

#### Exceptions:

***runtime\_error*** if the library is already initialized or an error occurs during initialization.

### 6.705.2.3 static void decaf::lang::Runtime::initializeRuntime (int *argc*, char \*\**argv*) [static]

Initialize the Decaf Library passing it the args that were passed to the application at startup.

#### Parameters:

*argc* - The number of args passed

*argv* - Array of char\* values passed to the Process on start.

#### Exceptions:

*runtime\_error* if the library is already initialized or an error occurs during initialization.

### 6.705.2.4 static void decaf::lang::Runtime::shutdownRuntime () [static]

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

#### Exceptions:

*runtime\_error* if the library has not already been initialized or an error occurs during shutdown.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runtime.h**

## 6.706 decaf::lang::exceptions::RuntimeException Class Reference

#include <src/main/decaf/lang/exceptions/RuntimeException.h> Inheritance diagram for decaf::lang::exceptions::RuntimeException:

### Public Member Functions

- **RuntimeException** () throw ()  
*Default Constructor.*
- **RuntimeException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other ActiveMQException.*
- **RuntimeException** (const **RuntimeException** &ex) throw ()  
*Copy Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **RuntimeException** (const std::exception \*cause) throw ()  
*Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RuntimeException** \* clone () const  
*Clones this exception.*
- virtual ~**RuntimeException** () throw ()

### 6.706.1 Constructor & Destructor Documentation

#### 6.706.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException () throw () [inline]

Default Constructor.

#### 6.706.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const Exception & ex) throw () [inline]

Conversion Constructor from some other ActiveMQException.

#### Parameters:

*ex* The **Exception** (p.1831) whose data is to be copied into this one.



### 6.706.1.3 decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* The **Exception** (p.1831) whose data is to be copied into this one.

### 6.706.1.4 decaf::lang::exceptions::RuntimeException::RuntimeException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.706.1.5 decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* **Pointer** (p.2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.706.1.6 decaf::lang::exceptions::RuntimeException::RuntimeException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.706.1.7** `virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException  
() throw () [inline, virtual]`

## **6.706.2 Member Function Documentation**

**6.706.2.1** `virtual RuntimeException* de-  
caf::lang::exceptions::RuntimeException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

an new **Exception** (p.1831) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/RuntimeException.h`

## 6.707 decaf::security::SecureRandom Class Reference

#include <src/main/decaf/security/SecureRandom.h> Inheritance diagram for decaf::security::SecureRandom:

### Public Member Functions

- **SecureRandom** ()  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- **SecureRandom** (const std::vector< unsigned char > &seed)  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- **SecureRandom** (const unsigned char \*seed, int size)  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- virtual ~**SecureRandom** ()
- virtual void **nextBytes** (std::vector< unsigned char > &buf)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*  
**Parameters:**  
*buf non-null array to contain the new random bytes*  
**See also:**  
*next* (p. 3156)
- virtual void **nextBytes** (unsigned char \*buf, int size)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*  
**Parameters:**  
*buf non-null array to contain the new random bytes*  
**See also:**  
*next* (p. 3156)  
**Exceptions:**  
***NullPointerException** if buff is NULL*  
***IllegalArgumentException** if size is negative*
- virtual void **setSeed** (unsigned long long seed)  
*Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.*  
**Parameters:**  
*seed the seed that alters the state of the random number generator*  
**See also:**  
*next* (p. 3156)  
***Random()*** (p. 3156)  
***Random(long)***

- virtual void **setSeed** (const std::vector< unsigned char > &seed)  
*Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.*
- virtual void **setSeed** (const unsigned char \*seed, int size)  
*Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.*

## Protected Member Functions

- virtual int **next** (int bits)  
*Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.*  
**Returns:**  
*int a pseudo-random generated int number*  
**Parameters:**  
*bits number of bits of the returned value*  
**See also:**  
*nextBytes (p. 3157)*  
*nextDouble (p. 3158)*  
*nextFloat (p. 3158)*  
*nextInt() (p. 3159)*  
*nextInt(int) (p. 3158)*  
*nextGaussian (p. 3158)*  
*nextLong (p. 3159)*

### 6.707.1 Detailed Description

Since:

1.0

### 6.707.2 Constructor & Destructor Documentation

#### 6.707.2.1 decaf::security::SecureRandom::SecureRandom ()

Creates a new instance of a secure random number generator that implements the default random number algorithm. The **SecureRandom** (p. 3331) instance that is created with this constructor is unseeded and can be seeded by calling the setSeed method. Calls to nextBytes on an unseeded **SecureRandom** (p. 3331) result in the object seeding itself.

#### 6.707.2.2 decaf::security::SecureRandom::SecureRandom (const std::vector< unsigned char > & seed)

Creates a new instance of a secure random number generator that implements the default random number algorithm. The **SecureRandom** (p. 3331) instance created by this constructor is seeded using the passed byte array.

**Parameters:**

*seed* The seed bytes to use to seed this secure random number generator.

**6.707.2.3 decaf::security::SecureRandom::SecureRandom (const unsigned char \*  
seed, int size)**

Creates a new instance of a secure random number generator that implements the default random number algorithm. The **SecureRandom** (p. 3331) instance created by this constructor is seeded using the passed byte array.

**Parameters:**

*seed* The seed bytes to use to seed this secure random number generator.

*size* The number of bytes in the seed buffer.

**Exceptions:**

*NullPointerException* if the seed buffer is NULL.

*IllegalArgumentException* if the size value is negative.

**6.707.2.4 virtual decaf::security::SecureRandom::~~SecureRandom () [virtual]****6.707.3 Member Function Documentation****6.707.3.1 virtual int decaf::security::SecureRandom::next (int bits) [protected,  
virtual]**

Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument *bits* as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

**Returns:**

int a pseudo-random generated int number

**Parameters:**

*bits* number of bits of the returned value

**See also:**

**nextBytes** (p. 3157)  
**nextDouble** (p. 3158)  
**nextFloat** (p. 3158)  
**nextInt()** (p. 3159)  
**nextInt(int)** (p. 3158)  
**nextGaussian** (p. 3158)  
**nextLong** (p. 3159)

Reimplemented from **decaf::util::Random** (p. 3156).

### 6.707.3.2 virtual void decaf::security::SecureRandom::nextBytes (unsigned char \* *buf*, int *size*) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters:

*buf* non-null array to contain the new random bytes

#### See also:

`next` (p. 3156)

#### Exceptions:

*NullPointerException* if *buf* is NULL

*IllegalArgumentException* if *size* is negative

Reimplemented from `decaf::util::Random` (p. 3157).

### 6.707.3.3 virtual void decaf::security::SecureRandom::nextBytes (std::vector< unsigned char > & *buf*) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters:

*buf* non-null array to contain the new random bytes

#### See also:

`next` (p. 3156)

Reimplemented from `decaf::util::Random` (p. 3157).

### 6.707.3.4 virtual void decaf::security::SecureRandom::setSeed (const unsigned char \* *seed*, int *size*) [virtual]

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

#### Parameters:

*seed* The seed bytes to use to seed this secure random number generator.

*size* The number of bytes in the seed buffer.

#### Exceptions:

*NullPointerException* if the seed buffer is NULL.

*IllegalArgumentException* if the size value is negative.

**6.707.3.5 virtual void decaf::security::SecureRandom::setSeed (const std::vector< unsigned char > & seed) [virtual]**

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

**Parameters:**

*seed* A vector of bytes that is used update the seed of the RNG.

**6.707.3.6 virtual void decaf::security::SecureRandom::setSeed (unsigned long long seed) [virtual]**

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

**Parameters:**

*seed* the seed that alters the state of the random number generator

**See also:**

**next** (p. 3156)  
**Random()** (p. 3156)  
**Random(long)**

Reimplemented from **decaf::util::Random** (p. 3159).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandom.h**

## 6.708 decaf::internal::security::SecureRandomImpl Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

#include <src/main/decaf/internal/security/unix/SecureRandomImpl.h> Inheritance diagram for decaf::internal::security::SecureRandomImpl:

### Public Member Functions

- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*
- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.708.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources. Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

**Since:**

1.0



## 6.708.2 Constructor & Destructor Documentation

**6.708.2.1** decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()

**6.708.2.2** virtual  
decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ()  
[virtual]

**6.708.2.3** decaf::internal::security::SecureRandomImpl::SecureRandomImpl ()

**6.708.2.4** virtual  
decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ()  
[virtual]

## 6.708.3 Member Function Documentation

**6.708.3.1** virtual unsigned char\* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator. The caller owns the returned array and must delete it.

### Parameters:

*numBytes* The number of bytes that should be generated for the new seed array.

Implements decaf::security::SecureRandomSpi (p. 3339).

**6.708.3.2** virtual unsigned char\* decaf::internal::security::SecureRandomImpl::providerGenerateSeed (int *numBytes*) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator. The caller owns the returned array and must delete it.

### Parameters:

*numBytes* The number of bytes that should be generated for the new seed array.

Implements decaf::security::SecureRandomSpi (p. 3339).

**6.708.3.3** virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char \* *bytes*, int *numBytes*) [virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array. The array must have already been allocated and be of the correct size to prevent segmentation faults.

### Parameters:

*bytes* The array that will be filled with random bytes equal to size.

*numBytes* The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3340).

**6.708.3.4 virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes (unsigned char \* *bytes*, int *numBytes*) [virtual]**

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array. The array must have already been allocated and be of the correct size to prevent segmentation faults.

**Parameters:**

*bytes* The array that will be filled with random bytes equal to size.

*numBytes* The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3340).

**6.708.3.5 virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char \* *seed*, int *size*) [virtual]**

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

**Parameters:**

*seed* The array of bytes used to update the generators seed.

*size* The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3340).

**6.708.3.6 virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed (const unsigned char \* *seed*, int *size*) [virtual]**

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

**Parameters:**

*seed* The array of bytes used to update the generators seed.

*size* The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3340).

The documentation for this class was generated from the following files:

- src/main/decaf/internal/security/unix/**SecureRandomImpl.h**
- src/main/decaf/internal/security/windows/**SecureRandomImpl.h**

## 6.709 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

#include <src/main/decaf/security/SecureRandomSpi.h> Inheritance diagram for decaf::security::SecureRandomSpi:

### Public Member Functions

- **SecureRandomSpi** ()
- virtual **~SecureRandomSpi** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)=0  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)=0  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)=0  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.709.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

Since:

1.0

### 6.709.2 Constructor & Destructor Documentation

**6.709.2.1** decaf::security::SecureRandomSpi::SecureRandomSpi ()

**6.709.2.2** virtual decaf::security::SecureRandomSpi::~~SecureRandomSpi ()  
 [virtual]

### 6.709.3 Member Function Documentation

**6.709.3.1** virtual unsigned char\* decaf::security::SecureRandomSpi::providerGenerateSeed (int *numBytes*) [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator. The caller owns the returned array and must delete it.

Parameters:

*numBytes* The number of bytes that should be generated for the new seed array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3337), and `decaf::internal::security::SecureRandomImpl` (p. 3337).

#### 6.709.3.2 `virtual void decaf::security::SecureRandomSpi::providerNextBytes (unsigned char * bytes, int numBytes)` [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array. The array must have already been allocated and be of the correct size to prevent segmentation faults.

##### Parameters:

*bytes* The array that will be filled with random bytes equal to size.

*numBytes* The number of bytes to generate and write into the bytes array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3338), and `decaf::internal::security::SecureRandomImpl` (p. 3338).

#### 6.709.3.3 `virtual void decaf::security::SecureRandomSpi::providerSetSeed (const unsigned char * seed, int size)` [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

##### Parameters:

*seed* The array of bytes used to update the generators seed.

*size* The size of the passed byte array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3338), and `decaf::internal::security::SecureRandomImpl` (p. 3338).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SecureRandomSpi.h`

## 6.710 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

### Public Member Functions

- **Semaphore** (int permits)  
*Creates a **Semaphore** (p. 3341) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)  
*Creates a **Semaphore** (p. 3341) with the given number of permits and the given fairness setting.*
- virtual **~Semaphore** ()
- void **acquire** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.*
- void **acquireUninterruptibly** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available.*
- bool **tryAcquire** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, only if one is available at the time of invocation.*
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.*
- void **release** () throw ( decaf::lang::exceptions::RuntimeException )  
*Releases a permit, returning it to the semaphore.*
- void **acquire** (int permits) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.*
- void **acquireUninterruptibly** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available.*
- bool **tryAcquire** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.*
- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )

*Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.*

- void **release** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )

*Releases the given number of permits, returning them to the semaphore.*

- int **availablePermits** () const

*Returns the current number of permits available in this semaphore.*

- int **drainPermits** () throw ( decaf::lang::exceptions::RuntimeException )

*Acquires and returns all permits that are immediately available.*

- bool **isFair** () const

- std::string **toString** () const

*Returns a string identifying this semaphore, as well as its state.*

## 6.710.1 Detailed Description

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 3344) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 3347) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 3341) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
```

```
static const int MAX_AVAILABLE = 100; Semaphore (p. 3341) available;
```

```
std::vector<std::string> items; std::vector<bool> used;
```

```
Mutex (p. 2782) lock;
```

```
public:
```

```
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.resize( MAX_AVAILABLE ); }
```

```
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvailableItem(); }
```

```
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
```

```
std::string getNextAvailableItem() {
```

```
synchronized( &lock ) (p. 4893) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i] ) { used[i] = true; return items[i]; } }
```

```
return std::string(); // not reached }
```

```
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 4893) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false; return true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back

to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 3344) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 2375) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 3344) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific **internal** (p. 144) points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

**Since:**

1.0

## 6.710.2 Constructor & Destructor Documentation

### 6.710.2.1 decaf::util::concurrent::Semaphore::Semaphore (int *permits*)

Creates a **Semaphore** (p. 3341) with the given number of permits and nonfair fairness setting.

**Parameters:**

*permits* the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

### 6.710.2.2 decaf::util::concurrent::Semaphore::Semaphore (int *permits*, bool *fair*)

Creates a **Semaphore** (p. 3341) with the given number of permits and the given fairness setting.

**Parameters:**

*permits* the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

*fair* true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

**6.710.2.3** `virtual decaf::util::concurrent::Semaphore::~~Semaphore ()` [virtual]

### 6.710.3 Member Function Documentation

**6.710.3.1** `void decaf::util::concurrent::Semaphore::acquire (int permits)  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted. Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to `release()` (p. 3347).

#### Parameters:

*permits* the number of permits to acquire.

#### Exceptions:

*InterruptedException* if the current thread is interrupted.

*IllegalArgumentException* if the permits argument is negative.

*RuntimeException* if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.2** `void decaf::util::concurrent::Semaphore::acquire ()  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted. Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes the `release()` (p. 3347) method for this semaphore and the current thread is next to be assigned a permit; or \* Some other thread interrupts the current thread.



If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

**Exceptions:**

**InterruptedException** - if the current thread is interrupted.

**RuntimeException** if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int *permits*) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )**

Acquires the given number of permits from this semaphore, blocking until all are available. Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

**Parameters:**

*permits* the number of permits to acquire.

**Exceptions:**

**IllegalArgumentException** if the permits argument is negative.

**RuntimeException** if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly () throw ( decaf::lang::exceptions::RuntimeException )**

Acquires a permit from this semaphore, blocking until one is available. Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 3347) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

**Exceptions:**

**RuntimeException** if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.5   int decaf::util::concurrent::Semaphore::availablePermits () const**

Returns the current number of permits available in this semaphore. This method is typically used for debugging and testing purposes.

**Returns:**

the number of permits available in this semaphore

**6.710.3.6   int decaf::util::concurrent::Semaphore::drainPermits () throw ( decaf::lang::exceptions::RuntimeException )**

Acquires and returns all permits that are immediately available.

**Returns:**

the number of permits acquired

**Exceptions:**

*RuntimeException* if an unexpected error occurs while draining the **Semaphore** (p. 3341).

**6.710.3.7   bool decaf::util::concurrent::Semaphore::isFair () const****Returns:**

true if this semaphore has fairness set true

**6.710.3.8   void decaf::util::concurrent::Semaphore::release (int *permits*)  
              throw ( decaf::lang::exceptions::IllegalArgumentException,  
                      decaf::lang::exceptions::RuntimeException )**

Releases the given number of permits, returning them to the semaphore. Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

**Parameters:**

*permits* the number of permits to release

**Exceptions:**

*IllegalArgumentException* if the permits argument is negative.

*RuntimeException* if an unexpected error occurs while releasing the **Semaphore** (p. 3341).

### 6.710.3.9 void decaf::util::concurrent::Semaphore::release () throw ( decaf::lang::exceptions::RuntimeException )

Releases a permit, returning it to the semaphore. Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 3344). Correct usage of a semaphore is established by programming convention in the application.

#### Exceptions:

*RuntimeException* if an unexpected error occurs while releasing the **Semaphore** (p. 3341).

### 6.710.3.10 std::string decaf::util::concurrent::Semaphore::toString () const

Returns a string identifying this semaphore, as well as its state. The state, in brackets, includes the String "Permits =" followed by the number of permits.

#### Returns:

a string identifying this semaphore, as well as its state

### 6.710.3.11 bool decaf::util::concurrent::Semaphore::tryAcquire (int *permits*, long long *timeout*, const TimeUnit & *unit*) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted. Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3347).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3347).

**Parameters:**

*permits* the number of permits to acquire  
*timeout* the maximum amount of time to wait to acquire the permits.  
*unit* the units that the timeout param represents.

**Returns:**

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

**Exceptions:**

*IllegalArgumentException* if the permits argument is negative.  
*RuntimeException* if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.12** `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits)  
 throw ( decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation. Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(permits, 0, **TimeUnit.SECONDS** (p. 3816)) which is almost equivalent (it also detects interruption).

**Parameters:**

*permits* the number of permits to acquire

**Returns:**

true if the permits were acquired and false otherwise.

**Exceptions:**

*IllegalArgumentException* if the permits argument is negative.  
*RuntimeException* if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

**6.710.3.13** `bool decaf::util::concurrent::Semaphore::tryAcquire  
 (long long timeout, const TimeUnit & unit) throw  
 ( decaf::lang::exceptions::InterruptedException,  
 decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted. Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- \* Some other thread invokes the **release()** (p. 3347) method for this semaphore and the current thread is next to be assigned a permit; or
- \* Some other thread interrupts the current thread; or
- \* The specified waiting time elapses.

If a permit is acquired then the value `true` is returned.

If the current thread:

- \* has its interrupted status set on entry to this method; or
- \* is interrupted while waiting to acquire a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value `false` is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters:

*timeout* the maximum time to wait for a permit

*unit* the time unit of the timeout argument

#### Returns:

`true` if a permit was acquired and `false` if the waiting time elapsed before a permit was acquired

#### Exceptions:

*InterruptedException* if the current thread is interrupted.

*RuntimeException* if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

#### 6.710.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire () throw ( decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, only if one is available at the time of invocation. Acquires a permit, if one is available and returns immediately, with the value `true`, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value `false`.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 3349) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use `tryAcquire(0, TimeUnit.SECONDS)` (p. 3816)) which is almost equivalent (it also detects interruption).

#### Returns:

`true` if a permit was acquired and `false` otherwise

#### Exceptions:

*RuntimeException* if an unexpected error occurs while acquiring the **Semaphore** (p. 3341).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Semaphore.h`

## 6.711 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

### Public Member Functions

- **SendExecutor** (**MessageCreator** \*messageCreator, **CmsTemplate** \*parent)
- virtual **~SendExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session, **cms::MessageProducer** \*producer) throw (**cms::CMSEException** )

*Execute an action given a session and producer.*

### Protected Member Functions

- **SendExecutor** (const **SendExecutor** &)
- **SendExecutor** & operator= (const **SendExecutor** &)

### 6.711.1 Constructor & Destructor Documentation

- 6.711.1.1 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** (const **SendExecutor** &) [inline, protected]
- 6.711.1.2 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** (**MessageCreator** \* *messageCreator*, **CmsTemplate** \* *parent*) [inline]
- 6.711.1.3 virtual **activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor** () [inline, virtual]

### 6.711.2 Member Function Documentation

- 6.711.2.1 virtual void **activemq::cmsutil::CmsTemplate::SendExecutor::doInCms** (**cms::Session** \* *session*, **cms::MessageProducer** \* *producer*) throw (**cms::CMSEException** ) [inline, virtual]

Execute an action given a session and producer.

#### Parameters:

*session* the CMS Session

*producer* the CMS Producer

#### Exceptions:

*cms::CMSEException* (p. 1160) if thrown by CMS API methods

Implements **activemq::cmsutil::ProducerCallback** (p. 3064).

### 6.711.2.2 SendExecutor& activemq::cmsutil::CmsTemplate::SendExecutor::operator=(const SendExecutor &) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.712 decaf::net::ServerSocket Class Reference

This class implements server sockets.

#include <src/main/decaf/net/ServerSocket.h> Inheritance diagram for decaf::net::ServerSocket:

### Public Member Functions

- **ServerSocket** ()  
*Creates a non-bound server socket.*
- **ServerSocket** (int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog, const **InetAddress** \*address) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- virtual ~**ServerSocket** ()  
*Releases socket handle if **close()** (p. 3357) hasn't been called.*
- virtual void **bind** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*
- virtual void **bind** (const std::string &host, int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*
- virtual **Socket** \* **accept** () throw ( decaf::io::IOException )  
*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3352), the caller blocks until a connection is made.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.*
- virtual bool **isClosed** () const
- virtual bool **isBound** () const
- virtual int **getReceiveBufferSize** () const throw ( SocketException )



*Gets the receive buffer size for this socket, SO\_RCVBUF.*

- virtual void **setReceiveBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Sets the receive buffer size for this socket, SO\_RCVBUF.*

- virtual bool **getReuseAddress** () const throw ( SocketException )

*Gets the reuse address flag, SO\_REUSEADDR.*

- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )

*Sets the reuse address flag, SO\_REUSEADDR.*

- virtual int **getSoTimeout** () const throw ( SocketException )

*Gets the timeout for socket operations, SO\_TIMEOUT.*

- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Sets the timeout for socket operations, SO\_TIMEOUT.*

- virtual int **getLocalPort** () const

*Gets the port number on the Local machine that this **ServerSocket** (p. 3352) is bound to.*

- virtual std::string **toString** () const

## Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )

*Sets the instance of a **SocketImplFactory** (p. 3537) that the **ServerSocket** (p. 3352) class should use when new instances of this class are created.*

## Protected Member Functions

- **ServerSocket** (SocketImpl \*impl)

*Creates a **ServerSocket** (p. 3352) wrapping the provided **SocketImpl** (p. 3529) instance, this **Socket** (p. 3503) is considered unconnected.*

- virtual void **implAccept** (Socket \*socket) throw ( decaf::io::IOException )

*Virtual method that allows a **ServerSocket** (p. 3352) subclass to override the accept call and provide its own **SocketImpl** (p. 3529) for the socket.*

- virtual int **getDefaultBacklog** ()

*Allows a subclass to override what is considered the default backlog.*

- void **checkClosed** () const throw ( decaf::io::IOException )
- void **ensureCreated** () const throw ( decaf::io::IOException )
- void **setupSocketImpl** (int port, int backlog, const **InetAddress** \*ifAddress)

### 6.712.1 Detailed Description

This class implements server sockets. A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 3529) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

#### Since:

1.0

### 6.712.2 Constructor & Destructor Documentation

#### 6.712.2.1 `decaf::net::ServerSocket::ServerSocket ()`

Creates a non-bound server socket.

#### 6.712.2.2 `decaf::net::ServerSocket::ServerSocket (int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )`

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3537) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

#### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

#### Exceptions:

*IOException* if there is an I/O error while performing this operation.

*IllegalArgumentException* if the port value is negative or greater than 65535.

#### 6.712.2.3 `decaf::net::ServerSocket::ServerSocket (int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )`

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3537) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

#### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The the number of incoming connection attempts to queue before connections are refused.

**Exceptions:**

*IOException* if there is an I/O error while performing this operation.

*IllegalArgumentException* if the port value is negative or greater than 65535.

#### 6.712.2.4 decaf::net::ServerSocket::ServerSocket (int *port*, int *backlog*, const InetAddress \* *address*) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3537) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

**Parameters:**

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The the number of incoming connection attempts to queue before connections are refused.

*ifAddress* The IP Address to bind to on the local machine.

**Exceptions:**

*IOException* if there is an I/O error while performing this operation.

*IllegalArgumentException* if the port value is negative or greater than 65535.

#### 6.712.2.5 virtual decaf::net::ServerSocket::~~ServerSocket () [virtual]

Releases socket handle if **close()** (p. 3357) hasn't been called.

#### 6.712.2.6 decaf::net::ServerSocket::ServerSocket (SocketImpl \* *impl*) [protected]

Creates a **ServerSocket** (p. 3352) wrapping the provided **SocketImpl** (p. 3529) instance, this **Socket** (p. 3503) is considered unconnected. The **ServerSocket** (p. 3352) class takes ownership of this **SocketImpl** (p. 3529) pointer and will delete it when the **Socket** (p. 3503) class is destroyed.

**Parameters:**

*impl* The **SocketImpl** (p. 3529) instance to wrap.

**Exceptions:**

*NullPointerException* if the passed **SocketImpl** (p. 3529) is Null.

### 6.712.3 Member Function Documentation

#### 6.712.3.1 virtual Socket\* decaf::net::ServerSocket::accept () throw ( decaf::io::IOException ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3352), the caller blocks until a connection is made. If the SO\_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3542) if the operation times out.

##### Returns:

a new **Socket** (p. 3503) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

##### Exceptions:

**IOException** if an I/O error occurs while binding the socket.

**SocketException** (p. 3521) if an error occurs while blocking on the accept call.

**SocketTimeoutException** (p. 3542) if the SO\_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2849).

#### 6.712.3.2 virtual void decaf::net::ServerSocket::bind (const std::string & host, int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen. If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

##### Parameters:

**host** The IP address or host name.

**port** The TCP port between 1..65535.

**backlog** The size of listen backlog.

##### Exceptions:

**IOException** if an I/O error occurs while binding the socket.

**IllegalArgumentException** if the parameters are not valid.

#### 6.712.3.3 virtual void decaf::net::ServerSocket::bind (const std::string & host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

##### Parameters:

**host** The IP address or host name.

*port* The TCP port between 1..65535.

**Exceptions:**

*IOException* if an I/O error occurs while binding the socket.

*IllegalArgumentException* if the parameters are not valid.

**6.712.3.4** void decaf::net::ServerSocket::checkClosed () const throw ( decaf::io::IOException ) [protected]

**6.712.3.5** virtual void decaf::net::ServerSocket::close () throw ( decaf::io::IOException ) [virtual]

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

**6.712.3.6** void decaf::net::ServerSocket::ensureCreated () const throw ( decaf::io::IOException ) [protected]

**6.712.3.7** virtual int decaf::net::ServerSocket::getDefaultBacklog () [protected, virtual]

Allows a subclass to override what is considered the default backlog.

**Returns:**

the default backlog for connections.

**6.712.3.8** virtual int decaf::net::ServerSocket::getLocalPort () const [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p. 3352) is bound to.

**Returns:**

the port number of this machine that is bound, if not bound returns -1.

**6.712.3.9** virtual int decaf::net::ServerSocket::getReceiveBufferSize () const throw ( SocketException ) [virtual]

Gets the receive buffer size for this socket, SO\_RCVBUF. This is the buffer used by the underlying platform socket to buffer received data.

**Returns:**

the receive buffer size in bytes.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.712.3.10** `virtual bool decaf::net::ServerSocket::getReuseAddress () const throw (SocketException ) [virtual]`

Gets the reuse address flag, SO\_REUSEADDR.

**Returns:**

True if the address can be reused.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.712.3.11** `virtual int decaf::net::ServerSocket::getSoTimeout () const throw (SocketException ) [virtual]`

Gets the timeout for socket operations, SO\_TIMEOUT.

**Returns:**

The timeout in milliseconds for socket operations.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to retrieve the information.

**6.712.3.12** `virtual void decaf::net::ServerSocket::implAccept (Socket * socket) throw ( decaf::io::IOException ) [protected, virtual]`

Virtual method that allows a **ServerSocket** (p. 3352) subclass to override the accept call and provide its own **SocketImpl** (p. 3529) for the socket.

**Parameters:**

*socket* The socket object whose **SocketImpl** (p. 3529) should be used for the accept call.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

**6.712.3.13** `virtual bool decaf::net::ServerSocket::isBound () const [virtual]`

**Returns:**

true if the server socket is bound.

**6.712.3.14** `virtual bool decaf::net::ServerSocket::isClosed () const [virtual]`

**Returns:**

true if the close method has been called on the **ServerSocket** (p. 3352).

**6.712.3.15**    `virtual void decaf::net::ServerSocket::setReceiveBufferSize  
                  (int size) throw ( SocketException, de-  
                  caf::lang::exceptions::IllegalArgumentException )  
                  [virtual]`

Sets the receive buffer size for this socket, SO\_RCVBUF.

**Parameters:**

*size*    Number of bytes to set the receive buffer to.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

*IllegalArgumentException* if the value is zero or negative.

**6.712.3.16**    `virtual void decaf::net::ServerSocket::setReuseAddress (bool reuse)  
                  throw ( SocketException ) [virtual]`

Sets the reuse address flag, SO\_REUSEADDR.

**Parameters:**

*reuse*    If true, sets the flag.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.712.3.17**    `static void decaf::net::ServerSocket::setSocketImplFactory  
                  (SocketImplFactory * factory) throw ( decaf::io::IOException,  
                  decaf::net::SocketException ) [static]`

Sets the instance of a **SocketImplFactory** (p. 3537) that the **ServerSocket** (p. 3352) class should use when new instances of this class are created. This method is only allowed to be used once during the lifetime of the application.

**Parameters:**

*factory*    The instance of a **SocketImplFactory** (p. 3537) to use when new **SocketImpl** (p. 3529) objects are created.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

*SocketException* (p. 3521) if this method has already been called with a valid factory.

**6.712.3.18**    `virtual void decaf::net::ServerSocket::setSoTimeout (int timeout) throw  
                  ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
                  [virtual]`

Sets the timeout for socket operations, SO\_TIMEOUT. A value of zero indicates that timeout is infinite for operations on this socket.

**Parameters:**

*timeout* The timeout in milliseconds for socket operations.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to set the information.

*IllegalArgumentException* if the timeout value is negative.

**6.712.3.19**    `void decaf::net::ServerSocket::setupSocketImpl (int port, int backlog,  
                  const InetAddress * ifAddress)` [protected]

**6.712.3.20**    `virtual std::string decaf::net::ServerSocket::toString () const` [virtual]

**Returns:**

a string representing this **ServerSocket** (p. 3352).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`



## 6.713 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

#include <src/main/decaf/net/ServerSocketFactory.h>Inheritance diagram for decaf::net::ServerSocketFactory:

### Public Member Functions

- virtual `~ServerSocketFactory ()`
- virtual `ServerSocket * createServerSocket ()`  
*Create a new **ServerSocket** (p. 3352) that is unbound.*
- virtual `ServerSocket * createServerSocket (int port)=0`  
*Create a new **ServerSocket** (p. 3352) that is bound to the given port.*
- virtual `ServerSocket * createServerSocket (int port, int backlog)=0`  
*Create a new **ServerSocket** (p. 3352) that is bound to the given port.*
- virtual `ServerSocket * createServerSocket (int port, int backlog, const InetAddress *address)=0`  
*Create a new **ServerSocket** (p. 3352) that is bound to the given port.*

### Static Public Member Functions

- static `ServerSocketFactory * getDefault ()`  
*Returns the Default **ServerSocket** (p. 3352) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.*

### Protected Member Functions

- `ServerSocketFactory ()`

#### 6.713.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

Since:

1.0

## 6.713.2 Constructor & Destructor Documentation

**6.713.2.1** `decaf::net::ServerSocketFactory::ServerSocketFactory ()` [protected]

**6.713.2.2** `virtual decaf::net::ServerSocketFactory::~~ServerSocketFactory ()`  
[virtual]

## 6.713.3 Member Function Documentation

**6.713.3.1** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int port, int backlog, const InetAddress * address)` [pure virtual]

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3352) will listen on all interfaces.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

*address* The address of the interface on the local machine to bind to.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1683), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1694), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2855).

**6.713.3.2** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket (int port, int backlog)` [pure virtual]

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3352) will use the specified connection backlog setting.

### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The number of pending connect request the **ServerSocket** (p. 3352) can queue.

### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1684), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1694), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2855).

#### 6.713.3.3 virtual ServerSocket\* decaf::net::ServerSocketFactory::createServerSocket(int *port*) [pure virtual]

Create a new **ServerSocket** (p. 3352) that is bound to the given port. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

##### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

##### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

##### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1684), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1695), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2856).

#### 6.713.3.4 virtual ServerSocket\* decaf::net::ServerSocketFactory::createServerSocket() [virtual]

Create a new **ServerSocket** (p. 3352) that is unbound. The **ServerSocket** (p. 3352) will have been configured with the defaults from the factory.

##### Returns:

new **ServerSocket** (p. 3352) pointer that is owned by the caller.

##### Exceptions:

*IOException* if the **ServerSocket** (p. 3352) cannot be created for some reason.

Reimplemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1685), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1695), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2856).

#### 6.713.3.5 static ServerSocketFactory\* decaf::net::ServerSocketFactory::getDefault() [static]

Returns the Default **ServerSocket** (p. 3352) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller. Only one default **ServerSocketFactory** (p. 3361) exists for the lifetime of the Application.

##### Returns:

the default **ServerSocketFactory** (p. 3361) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 3561).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocketFactory.h`

## 6.714 cms::Session Class Reference

A **Session** (p.3365) object is a single-threaded context for producing and consuming messages.

#include <src/main/cms/Session.h> Inheritance diagram for cms::Session:

### Public Types

- enum **AcknowledgeMode** {  
**AUTO\_ACKNOWLEDGE**, **DUPS\_OK\_ACKNOWLEDGE**, **CLIENT\_-ACKNOWLEDGE**, **SESSION\_TRANSACTIONED**,  
**INDIVIDUAL\_ACKNOWLEDGE** }

### Public Member Functions

- virtual **~Session** ()
- virtual void **close** ()=0 throw ( CMSEException )  
*Closes this session as well as any active child consumers or producers.*
- virtual void **commit** ()=0 throw ( CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** ()=0 throw ( CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** ()=0 throw ( CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2589) for the specified destination.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2589) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector, bool noLocal)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2589) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createDurableConsumer** (const **Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw ( CMSEException )  
*Creates a durable subscriber to the specified topic, using a **Message** (p. 2534) selector.*
- virtual **MessageProducer** \* **createProducer** (const **Destination** \*destination)=0 throw ( CMSEException )

*Creates a **MessageProducer** (p. 2725) to send messages to the specified destination.*

- virtual **QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue)=0 throw ( CMSException )

*Creates a new **QueueBrowser** (p. 3153) to peek at Messages on the given **Queue** (p. 3148).*

- virtual **QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue, const std::string &selector)=0 throw ( CMSException )

*Creates a new **QueueBrowser** (p. 3153) to peek at Messages on the given **Queue** (p. 3148).*

- virtual **Queue** \* **createQueue** (const std::string &queueName)=0 throw ( CMSException )

*Creates a queue identity given a **Queue** (p. 3148) name.*

- virtual **Topic** \* **createTopic** (const std::string &topicName)=0 throw ( CMSException )

*Creates a topic identity given a **Queue** (p. 3148) name.*

- virtual **TemporaryQueue** \* **createTemporaryQueue** ()=0 throw ( CMSException )

*Creates a **TemporaryQueue** (p. 3759) object.*

- virtual **TemporaryTopic** \* **createTemporaryTopic** ()=0 throw ( CMSException )

*Creates a **TemporaryTopic** (p. 3761) object.*

- virtual **Message** \* **createMessage** ()=0 throw ( CMSException )

*Creates a new **Message** (p. 2534).*

- virtual **BytesMessage** \* **createBytesMessage** ()=0 throw ( CMSException )

*Creates a **BytesMessage** (p. 1056).*

- virtual **BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize)=0 throw ( CMSException )

*Creates a **BytesMessage** (p. 1056) and sets the payload to the passed value.*

- virtual **StreamMessage** \* **createStreamMessage** ()=0 throw ( CMSException )

*Creates a new **StreamMessage** (p. 3652).*

- virtual **TextMessage** \* **createTextMessage** ()=0 throw ( CMSException )

*Creates a new **TextMessage** (p. 3763).*

- virtual **TextMessage** \* **createTextMessage** (const std::string &text)=0 throw ( CMSException )

*Creates a new **TextMessage** (p. 3763) and set the text to the value given.*

- virtual **MapMessage** \* **createMapMessage** ()=0 throw ( CMSException )

*Creates a new **MapMessage** (p. 2472).*

- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw ( CMSException )

*Returns the acknowledgment mode of the session.*

- virtual bool **isTransacted** () const =0 throw ( CMSEException )  
*Gets if the Session is a Transacted **Session** (p. 3365).*
- virtual void **unsubscribe** (const std::string &name)=0 throw ( CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*

### 6.714.1 Detailed Description

A **Session** (p. 3365) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for TemporaryTopics and TemporaryQueues.
- It provides a way to create **Queue** (p. 3148) or **Topic** (p. 3817) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2589) until a message arrives. The thread may then use one or more of the Session's MessageProducers.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's **close** method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 3365) method that can be called concurrently.
- Invoking any other **Session** (p. 3365) method on a closed session must throw an **IllegalStateException** (p. 1997). Closing a closed session must not throw any exceptions.

#### Transacted Sessions

When a **Session** (p. 3365) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 3365) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 3365). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2725) this implies that all messages sent by the producer are not sent to the Provider until the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2589) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2534) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2534).

Since:

1.0

## 6.714.2 Member Enumeration Documentation

### 6.714.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

**AUTO\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

**DUPS\_OK\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

**CLIENT\_ACKNOWLEDGE** With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

**SESSION\_TRANSACTED** Messages will be consumed when the transaction commits.

**INDIVIDUAL\_ACKNOWLEDGE** **Message** (p. 2534) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

## 6.714.3 Constructor & Destructor Documentation

### 6.714.3.1 virtual cms::Session::~~Session () [inline, virtual]

## 6.714.4 Member Function Documentation

### 6.714.4.1 virtual void cms::Session::close () throw ( CMSEException ) [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions:

**CMSEException** (p. 1160) - If an internal error occurs.

Implements **cms::Closeable** (p. 1148).



Implemented in **activemq::cmsutil::PooledSession** (p. 2957), and **activemq::core::ActiveMQSession** (p. 520).

**6.714.4.2** `virtual void cms::Session::commit () throw ( CMSExcption ) [pure virtual]`

Commits all messages done in this transaction and releases any locks currently held.

**Exceptions:**

*CMSExcption* (p. 1160) - If an internal error occurs.

*IllegalStateException* (p. 1997) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2958), and **activemq::core::ActiveMQSession** (p. 520).

Referenced by **activemq::cmsutil::PooledSession::commit()**.

**6.714.4.3** `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue, const std::string & selector) throw ( CMSExcption ) [pure virtual]`

Creates a new **QueueBrowser** (p. 3153) to peek at Messages on the given **Queue** (p. 3148).

**Parameters:**

*queue* the **Queue** (p. 3148) to browse

*selector* the **Message** (p. 2534) selector to filter which messages are browsed.

**Returns:**

New **QueueBrowser** (p. 3153) that is owned by the caller.

**Exceptions:**

*CMSExcption* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2958), and **activemq::core::ActiveMQSession** (p. 520).

**6.714.4.4** `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue) throw ( CMSExcption ) [pure virtual]`

Creates a new **QueueBrowser** (p. 3153) to peek at Messages on the given **Queue** (p. 3148).

**Parameters:**

*queue* the **Queue** (p. 3148) to browse

**Returns:**

New **QueueBrowser** (p. 3153) that is owned by the caller.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2958), and **activemq::core::ActiveMQSession** (p. 521).

**6.714.4.5** `virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * bytes, int bytesSize) throw ( CMSEException) [pure virtual]`

Creates a **BytesMessage** (p. 1056) and sets the payload to the passed value.

**Parameters:**

*bytes* an array of bytes to set in the message

*bytesSize* the size of the bytes array, or number of bytes to use

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2959), and **activemq::core::ActiveMQSession** (p. 521).

**6.714.4.6** `virtual BytesMessage* cms::Session::createBytesMessage () throw ( CMSEException) [pure virtual]`

Creates a **BytesMessage** (p. 1056).

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2959), and **activemq::core::ActiveMQSession** (p. 521).

**6.714.4.7** `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector, bool noLocal) throw ( CMSEException) [pure virtual]`

Creates a **MessageConsumer** (p. 2589) for the specified destination, using a message selector.

**Parameters:**

*destination* the **Destination** (p. 1723) that this consumer receiving messages for.

*selector* the **Message** (p. 2534) Selector to use

*noLocal* if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns:**

pointer to a new **MessageConsumer** (p. 2589) that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if an invalid destination is specified.

*InvalidSelectorException* (p. 2138) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2960), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.8** virtual **MessageConsumer\*** **cms::Session::createConsumer** (**const** **Destination \*** *destination*, **const** **std::string &** *selector*) **throw** (**CMSEException**) [pure virtual]

Creates a **MessageConsumer** (p. 2589) for the specified destination, using a message selector.

**Parameters:**

*destination* the **Destination** (p. 1723) that this consumer receiving messages for.

*selector* the **Message** (p. 2534) Selector to use

**Returns:**

pointer to a new **MessageConsumer** (p. 2589) that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if an invalid destination is specified.

*InvalidSelectorException* (p. 2138) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2961), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.9** virtual **MessageConsumer\*** **cms::Session::createConsumer** (**const** **Destination \*** *destination*) **throw** (**CMSEException**) [pure virtual]

Creates a **MessageConsumer** (p. 2589) for the specified destination.

**Parameters:**

*destination* the **Destination** (p. 1723) that this consumer receiving messages for.

**Returns:**

pointer to a new **MessageConsumer** (p. 2589) that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2961), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.10** `virtual MessageConsumer* cms::Session::createDurableConsumer (const Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw ( CMSEException ) [pure virtual]`

Creates a durable subscriber to the specified topic, using a **Message** (p. 2534) selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

**Parameters:**

*destination* the topic to subscribe to

*name* The name used to identify the subscription

*selector* the **Message** (p. 2534) Selector to use

*noLocal* if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns:**

pointer to a new durable **MessageConsumer** (p. 2589) that is owned by the caller ( caller deletes )

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if an invalid destination is specified.

*InvalidSelectorException* (p. 2138) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2961), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.11** `virtual MapMessage* cms::Session::createMapMessage () throw ( CMSEException ) [pure virtual]`

Creates a new **MapMessage** (p. 2472).

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2962), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.12** `virtual Message* cms::Session::createMessage () throw ( CMSEException ) [pure virtual]`

Creates a new **Message** (p. 2534).

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2962), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.13** virtual MessageProducer\* cms::Session::createProducer (const Destination \* *destination*) throw ( CMSEException ) [pure virtual]

Creates a MessageProducer (p. 2725) to send messages to the specified destination.

**Parameters:**

*destination* the Destination (p. 1723) to send on

**Returns:**

New MessageProducer (p. 2725) that is owned by the caller.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

*InvalidDestinationException* (p. 2131) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2962), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.14** virtual Queue\* cms::Session::createQueue (const std::string & *queueName*) throw ( CMSEException ) [pure virtual]

Creates a queue identity given a Queue (p. 3148) name.

**Parameters:**

*queueName* the name of the new Queue (p. 3148)

**Returns:**

new Queue (p. 3148) pointer that is owned by the caller.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2963), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.15** virtual StreamMessage\* cms::Session::createStreamMessage () throw ( CMSEException ) [pure virtual]

Creates a new StreamMessage (p. 3652).

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2963), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.16** `virtual TemporaryQueue* cms::Session::createTemporaryQueue ()  
throw ( CMSEException ) [pure virtual]`

Creates a **TemporaryQueue** (p. 3759) object.

**Returns:**

new **TemporaryQueue** (p. 3759) pointer that is owned by the caller.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2963), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.17** `virtual TemporaryTopic* cms::Session::createTemporaryTopic () throw  
( CMSEException ) [pure virtual]`

Creates a **TemporaryTopic** (p. 3761) object.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2964), and **activemq::core::ActiveMQSession** (p. 525).

**6.714.4.18** `virtual TextMessage* cms::Session::createTextMessage (const std::string  
& text) throw ( CMSEException ) [pure virtual]`

Creates a new **TextMessage** (p. 3763) and set the text to the value given.

**Parameters:**

*text* the initial text for the message

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2964), and **activemq::core::ActiveMQSession** (p. 525).

**6.714.4.19** `virtual TextMessage* cms::Session::createTextMessage () throw (  
CMSEException ) [pure virtual]`

Creates a new **TextMessage** (p. 3763).

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2964), and **activemq::core::ActiveMQSession** (p. 525).

**6.714.4.20** `virtual Topic* cms::Session::createTopic (const std::string & topicName)  
throw ( CMSEException )` [pure virtual]

Creates a topic identity given a **Queue** (p. 3148) name.

**Parameters:**

*topicName* the name of the new **Topic** (p. 3817)

**Returns:**

new **Topic** (p. 3817) pointer that is owned by the caller.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2964), and **activemq::core::ActiveMQSession** (p. 526).

**6.714.4.21** `virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const  
throw ( CMSEException )` [pure virtual]

Returns the acknowledgment mode of the session.

**Returns:**

the Sessions Acknowledge Mode

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2965), and **activemq::core::ActiveMQSession** (p. 526).

**6.714.4.22** `virtual bool cms::Session::isTransacted () const throw ( CMSEException  
)` [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 3365).

**Returns:**

transacted true - false.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2965), and **activemq::core::ActiveMQSession** (p. 529).

#### 6.714.4.23 `virtual void cms::Session::recover () throw ( CMSEException ) [pure virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

#### Exceptions:

*CMSEException* (p. 1160) - if the CMS provider fails to stop and restart message delivery due to some internal error.

*IllegalStateException* (p. 1997) - if the method is called by a transacted session.

Implemented in `activemq::cmsutil::PooledSession` (p. 2966), and `activemq::core::ActiveMQSession` (p. 529).

#### 6.714.4.24 `virtual void cms::Session::rollback () throw ( CMSEException ) [pure virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

#### Exceptions:

*CMSEException* (p. 1160) - If an internal error occurs.

*IllegalStateException* (p. 1997) - if the method is not called by a transacted session.

Implemented in `activemq::cmsutil::PooledSession` (p. 2966), and `activemq::core::ActiveMQSession` (p. 530).

#### 6.714.4.25 `virtual void cms::Session::unsubscribe (const std::string & name) throw ( CMSEException ) [pure virtual]`

Unsubscribes a durable subscription that has been created by a client. This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active `MessageConsumer` (p. 2589) or `Subscriber` for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

#### Parameters:

*name* The name used to identify this subscription

#### Exceptions:

*CMSEException* (p. 1160) - If an internal error occurs.



Implemented in **activemq::cmsutil::PooledSession** (p. 2966), and **activemq::core::ActiveMQSession** (p. 532).

The documentation for this class was generated from the following file:

- `src/main/cms/Session.h`

## 6.715 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

#include <src/main/activemq/cmsutil/SessionCallback.h> Inheritance diagram for activemq::cmsutil::SessionCallback:

### Public Member Functions

- virtual `~SessionCallback ()`
- virtual void `doInCms (cms::Session *session)=0` throw ( cms::CMSEException )  
*Execute any number of operations against the supplied CMS session.*

#### 6.715.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

#### 6.715.2 Constructor & Destructor Documentation

- 6.715.2.1** virtual `activemq::cmsutil::SessionCallback::~~SessionCallback ()` [inline, virtual]

#### 6.715.3 Member Function Documentation

- 6.715.3.1** virtual void `activemq::cmsutil::SessionCallback::doInCms (cms::Session *session)` throw ( cms::CMSEException ) [pure virtual]

Execute any number of operations against the supplied CMS session.

#### Parameters:

*session* the CMS Session

#### Exceptions:

*cms::CMSEException* (p. 1160) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 3065), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 3175).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

## 6.716 activemq::commands::SessionId Class Reference

#include <src/main/activemq/commands/SessionId.h> Inheritance diagram for activemq::commands::SessionId:

### Public Types

- typedef decaf::lang::PointerComparator< SessionId > COMPARATOR

### Public Member Functions

- SessionId ()
- SessionId (const SessionId &other)
- SessionId (const ConnectionId \*connectionId, long long sessionId)
- SessionId (const ProducerId \*producerId)
- SessionId (const ConsumerId \*consumerId)
- virtual ~SessionId ()
- virtual unsigned char getDataStructureType () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual SessionId \* cloneDataStructure () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void copyDataStructure (const DataStructure \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string toString () const  
*Returns a string containing the information for this DataStructure (p. 1660) such as its type and value of its elements.*
- virtual bool equals (const DataStructure \*value) const  
*Compares the DataStructure (p. 1660) passed in to this one, and returns if they are equivalent.*
- const Pointer< ConnectionId > & getParentId () const
- virtual const std::string & getConnectionId () const
- virtual std::string & getConnectionId ()
- virtual void setConnectionId (const std::string &connectionId)
- virtual long long getValue () const
- virtual void setValue (long long value)
- virtual int compareTo (const SessionId &value) const
- virtual bool equals (const SessionId &value) const
- virtual bool operator== (const SessionId &value) const
- virtual bool operator< (const SessionId &value) const
- SessionId & operator= (const SessionId &other)

## Static Public Attributes

- static const unsigned char **ID\_SESSIONID** = 121

## Protected Attributes

- std::string **connectionId**
- long long **value**

## 6.716.1 Member Typedef Documentation

- 6.716.1.1**    `typedef decaf::lang::PointerComparator<SessionId>  
activemq::commands::SessionId::COMPARATOR`

## 6.716.2 Constructor & Destructor Documentation

- 6.716.2.1**    `activemq::commands::SessionId::SessionId ()`
- 6.716.2.2**    `activemq::commands::SessionId::SessionId (const SessionId & other)`
- 6.716.2.3**    `activemq::commands::SessionId::SessionId (const ConnectionId *  
connectionId, long long sessionId)`
- 6.716.2.4**    `activemq::commands::SessionId::SessionId (const ProducerId *  
producerId)`
- 6.716.2.5**    `activemq::commands::SessionId::SessionId (const ConsumerId *  
consumerId)`
- 6.716.2.6**    `virtual activemq::commands::SessionId::~~SessionId () [virtual]`

## 6.716.3 Member Function Documentation

- 6.716.3.1**    `virtual SessionId* activemq::commands::SessionId::cloneDataStructure ()  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

- 6.716.3.2**    `virtual int activemq::commands::SessionId::compareTo (const SessionId  
& value) const [virtual]`
- 6.716.3.3**    `virtual void activemq::commands::SessionId::copyDataStructure (const  
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

**6.716.3.4** virtual bool activemq::commands::SessionId::equals (const SessionId & *value*) const [virtual]

**6.716.3.5** virtual bool activemq::commands::SessionId::equals (const DataStructure \* *value*) const [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

**6.716.3.6** virtual std::string& activemq::commands::SessionId::getConnectionId () [virtual]

**6.716.3.7** virtual const std::string& activemq::commands::SessionId::getConnectionId () const [virtual]

**6.716.3.8** virtual unsigned char activemq::commands::SessionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.716.3.9** `const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId () const`
- 6.716.3.10** `virtual long long activemq::commands::SessionId::getValue () const [virtual]`
- 6.716.3.11** `virtual bool activemq::commands::SessionId::operator< (const SessionId & value) const [virtual]`
- 6.716.3.12** `SessionId& activemq::commands::SessionId::operator= (const SessionId & other)`
- 6.716.3.13** `virtual bool activemq::commands::SessionId::operator== (const SessionId & value) const [virtual]`
- 6.716.3.14** `virtual void activemq::commands::SessionId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.716.3.15** `virtual void activemq::commands::SessionId::setValue (long long value) [virtual]`
- 6.716.3.16** `virtual std::string activemq::commands::SessionId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p.833).

## 6.716.4 Field Documentation

- 6.716.4.1** `std::string activemq::commands::SessionId::connectionId [protected]`
- 6.716.4.2** `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.716.4.3** `long long activemq::commands::SessionId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3383).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.717.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3383). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.717.2 Constructor & Destructor Documentation

**6.717.2.1** `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.717.2.2** `virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.717.3 Member Function Documentation

**6.717.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.717.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.717.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.717.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.717.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.717.3.6** virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.717.3.7** virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h

## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3387).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.718.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3387). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.718.2 Constructor & Destructor Documentation

**6.718.2.1** `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.718.2.2** `virtual activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.718.3 Member Function Documentation

**6.718.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.718.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.718.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.718.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.718.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.718.3.6** virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.718.3.7** virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3391).

#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.719.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3391). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.719.2 Constructor & Destructor Documentation

**6.719.2.1** `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.719.2.2** `virtual activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.719.3 Member Function Documentation

**6.719.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.719.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.719.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.719.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.719.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.719.3.6** virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.719.3.7** virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h

## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3395).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.720.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3395). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.720.2 Constructor & Destructor Documentation

**6.720.2.1** `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.720.2.2** `virtual activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.720.3 Member Function Documentation

**6.720.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.720.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.720.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.720.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.720.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.720.3.6** virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.720.3.7** virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SessionIdMarshaller.h**

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3399).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.721.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3399). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.721.2 Constructor & Destructor Documentation

**6.721.2.1** `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.721.2.2** `virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.721.3 Member Function Documentation

**6.721.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.721.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.721.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.721.3.4** **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw (**  
**decaf::io::IOException ) [virtual]**

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.721.3.5** **virtual int ac-**  
**tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException**  
**) [virtual]**

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.721.3.6** virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.721.3.7** virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h

## 6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3403).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller:

### Public Member Functions

- `SessionIdMarshaller ()`
- `virtual ~SessionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.722.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionIdMarshaller` (p. 3403). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.722.2 Constructor & Destructor Documentation

**6.722.2.1** `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller()` [inline]

**6.722.2.2** `virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

## 6.722.3 Member Function Documentation

**6.722.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.722.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.722.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.722.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.722.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.722.3.6** virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.722.3.7** virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h**

## 6.723 activemq::commands::SessionInfo Class Reference

#include <src/main/activemq/commands/SessionInfo.h> Inheritance diagram for activemq::commands::SessionInfo:

### Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SessionInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< SessionId > & getSessionId** () const
- virtual **Pointer< SessionId > & getSessionId** ()
- virtual void **setSessionId** (const **Pointer< SessionId > &sessionId**)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_SESSIONINFO** = 4

### Protected Attributes

- **Pointer< SessionId > sessionId**

## 6.723.1 Constructor & Destructor Documentation

**6.723.1.1** `activemq::commands::SessionInfo::SessionInfo ()`

**6.723.1.2** `virtual activemq::commands::SessionInfo::~~SessionInfo ()` [virtual]

## 6.723.2 Member Function Documentation

**6.723.2.1** `virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.723.2.2** `virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.723.2.3** `Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand ()` `const`

**6.723.2.4** `virtual bool activemq::commands::SessionInfo::equals (const DataStructure * value)` `const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).



**6.723.2.5**    `unsigned int activemq::commands::SessionInfo::getAckMode () const`  
                   [inline]

**6.723.2.6**    `virtual unsigned char ac-`  
                   `tivemq::commands::SessionInfo::getDataStructureType ()`  
                   `const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSet** (p. 1660) type copy.

Implements **activemq::commands::DataSet** (p. 1662).

**6.723.2.7**    `virtual Pointer<SessionId>& ac-`  
                   `tivemq::commands::SessionInfo::getSessionId ()`  
                   [virtual]

**6.723.2.8**    `virtual const Pointer<SessionId>& ac-`  
                   `tivemq::commands::SessionInfo::getSessionId () const`  
                   [virtual]

**6.723.2.9**    `void activemq::commands::SessionInfo::setAckMode (unsigned int mode)`  
                   [inline]

**6.723.2.10**    `virtual void activemq::commands::SessionInfo::setSessionId (const`  
                   `Pointer< SessionId > & sessionId)` [virtual]

**6.723.2.11**    `virtual std::string activemq::commands::SessionInfo::toString () const`  
                   [virtual]

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.723.2.12**    `virtual Pointer<Command> activemq::commands::SessionInfo::visit`  
                   `(activemq::state::CommandVisitor * visitor) throw (`  
                   `exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

### 6.723.3 Field Documentation

**6.723.3.1** `const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4` [static]

**6.723.3.2** `Pointer<SessionId> activemq::commands::SessionInfo::sessionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

## 6.724 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3411).

#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.724.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3411). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.724.2 Constructor & Destructor Documentation

**6.724.2.1** `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.724.2.2** `virtual activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.724.3 Member Function Documentation

**6.724.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.724.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.724.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.724.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.724.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.724.3.6** virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.724.3.7** virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h

## 6.725 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3415).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>Inheritance  
diagram for activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller:

### Public Member Functions

- `SessionInfoMarshaller ()`
- `virtual ~SessionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.725.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3415). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.725.2 Constructor & Destructor Documentation

**6.725.2.1** `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.725.2.2** `virtual activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.725.3 Member Function Documentation

**6.725.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.725.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.725.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.725.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.725.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.725.3.6** virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.725.3.7** virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h

## 6.726 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3419).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller:

### Public Member Functions

- `SessionInfoMarshaller ()`
- `virtual ~SessionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.726.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3419). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.726.2 Constructor & Destructor Documentation

**6.726.2.1** `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.726.2.2** `virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.726.3 Member Function Documentation

**6.726.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.726.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.726.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 780).

**6.726.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 781).

**6.726.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 782).

**6.726.3.6** virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.726.3.7** virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**

## 6.727 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3423).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.727.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3423). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.727.2 Constructor & Destructor Documentation

**6.727.2.1** `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.727.2.2** `virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.727.3 Member Function Documentation

**6.727.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.727.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.727.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.727.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.727.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.727.3.6** virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.727.3.7** virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h

## 6.728 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3427).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.728.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3427). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.728.2 Constructor & Destructor Documentation

**6.728.2.1** `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.728.2.2** `virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.728.3 Member Function Documentation

**6.728.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.728.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.728.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.728.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.728.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.728.3.6** virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.728.3.7** virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SessionInfoMarshaller.h**

## 6.729 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3431).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller:

### Public Member Functions

- `SessionInfoMarshaller ()`
- `virtual ~SessionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.729.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for `SessionInfoMarshaller` (p. 3431). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.729.2 Constructor & Destructor Documentation

**6.729.2.1** `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

**6.729.2.2** `virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

## 6.729.3 Member Function Documentation

**6.729.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.729.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.729.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).



Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.729.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.729.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.729.3.6** virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.729.3.7** virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h

## 6.730 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

### Public Member Functions

- **SessionPool** (**cms::Connection** \*connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** \*resourceLifecycleManager)  
*Constructs a session pool.*
- virtual **~SessionPool** ()  
*Destroys the pooled session objects, but not the underlying session resources.*
- virtual **PooledSession** \* **takeSession** () throw ( cms::CMSException )  
*Takes a session from the pool, creating one if necessary.*
- virtual void **returnSession** (**PooledSession** \*session)  
*Returns a session to the pool.*
- **ResourceLifecycleManager** \* **getResourceLifecycleManager** ()

### Protected Member Functions

- **SessionPool** (const **SessionPool** &)
- **SessionPool** & **operator=** (const **SessionPool** &)

#### 6.730.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided **ResourceLifecycleManager** (p. 3282), not by this pool. This class is thread-safe.

#### 6.730.2 Constructor & Destructor Documentation

- 6.730.2.1** **activemq::cmsutil::SessionPool::SessionPool** (const **SessionPool** &)  
[inline, protected]
- 6.730.2.2** **activemq::cmsutil::SessionPool::SessionPool** (**cms::Connection** \* connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** \* resourceLifecycleManager)

Constructs a session pool.

#### Parameters:

**connection** the connection to be used for creating all sessions.

**ackMode** the acknowledge mode to be used for all sessions

**resourceLifecycleManager** the object responsible for managing the lifecycle of any allocated **cms::Session** (p. 3365) resources.

**6.730.2.3** virtual `activemq::cmsutil::SessionPool::~~SessionPool()` [virtual]

Destroys the pooled session objects, but not the underlying session resources. That is the job of the `ResourceLifecycleManager` (p. 3282).

**6.730.3 Member Function Documentation****6.730.3.1** `ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager()` [inline]**6.730.3.2** `SessionPool& activemq::cmsutil::SessionPool::operator= (const SessionPool &)` [inline, protected]**6.730.3.3** virtual void `activemq::cmsutil::SessionPool::returnSession (PooledSession * session)` [virtual]

Returns a session to the pool.

**Parameters:**

*session* the session to be returned.

**6.730.3.4** virtual `PooledSession* activemq::cmsutil::SessionPool::takeSession ()` throw ( `cms::CMSEException` ) [virtual]

Takes a session from the pool, creating one if necessary.

**Returns:**

the pooled session object

**Exceptions:**

`cms::CMSEException` (p. 1160) if an error occurred

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionPool.h`

## 6.731 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

### Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual **~SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

## 6.731.1 Constructor & Destructor Documentation

**6.731.1.1** `activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & info)`

**6.731.1.2** `virtual activemq::state::SessionState::~~SessionState ()` [virtual]

## 6.731.2 Member Function Documentation

**6.731.2.1** `void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & info)` [inline]

**6.731.2.2** `void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & info)`

**6.731.2.3** `void activemq::state::SessionState::checkShutdown ()` const

**6.731.2.4** `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & id)` [inline]

**6.731.2.5** `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates ()` const [inline]

**6.731.2.6** `const Pointer<SessionInfo> activemq::state::SessionState::getInfo ()` const [inline]

**6.731.2.7** `Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & id)` [inline]

**6.731.2.8** `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates ()` const [inline]

**6.731.2.9** `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & id)` [inline]

**6.731.2.10** `Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & id)`

**6.731.2.11** `void activemq::state::SessionState::shutdown ()` [inline]

**6.731.2.12** `std::string activemq::state::SessionState::toString ()` const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

## 6.732 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

#include <src/main/decaf/util/Set.h> Inheritance diagram for decaf::util::Set< E >:

### Public Member Functions

- virtual `~Set ()`

#### 6.732.1 Detailed Description

`template<typename E> class decaf::util::Set< E >`

A collection that contains no duplicate elements. More formally, sets contain no pair of elements `e1` and `e2` such that `e1 == e2`, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since:

1.0

#### 6.732.2 Constructor & Destructor Documentation

**6.732.2.1** `template<typename E> virtual decaf::util::Set< E >::~~Set () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

## 6.733 decaf::lang::Short Class Reference

#include <src/main/decaf/lang/Short.h> Inheritance diagram for decaf::lang::Short:

### Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const  
*Compares this **Short** (p. 3440) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Short** &s) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const short &s) const  
*Compares this **Short** (p. 3440) instance with another.*
- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const short &s) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*



## Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3665) into a **Short** (p. 3440).*
- static short **reverseBytes** (short value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.*
- static short **parseShort** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed short in the radix specified by the second argument.*
- static short **parseShort** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal short.*
- static **Short valueOf** (short value)  
*Returns a **Short** (p. 3440) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Short** (p. 3440) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Short** (p. 3440) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

## Static Public Attributes

- static const int **SIZE** = 16  
*Size of this objects primitive type in bits.*
- static const short **MAX\_VALUE** = (short)0x7FFF  
*Max Value for this Object's primitive type.*
- static const short **MIN\_VALUE** = (short)0x8000  
*Max Value for this Object's primitive type.*

### 6.733.1 Constructor & Destructor Documentation

#### 6.733.1.1 decaf::lang::Short::Short (short value)

##### Parameters:

*value* - short to wrap

**6.733.1.2**   `decaf::lang::Short::Short (const std::string & value) throw (exceptions::NumberFormatException )`

**Parameters:**

*value* - string value to convert to short and wrap

**Exceptions:**

*NumberFormatException*

**6.733.1.3**   `virtual decaf::lang::Short::~~Short () [inline, virtual]`

## **6.733.2   Member Function Documentation**

**6.733.2.1**   `virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2835).

**6.733.2.2**   `virtual int decaf::lang::Short::compareTo (const short & s) const [virtual]`

Compares this **Short** (p. 3440) instance with another.

**Parameters:**

*s* - the **Short** (p. 3440) instance to be compared

**Returns:**

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< short >` (p. 1214).

**6.733.2.3**   `virtual int decaf::lang::Short::compareTo (const Short & s) const [virtual]`

Compares this **Short** (p. 3440) instance with another.

**Parameters:**

*s* - the **Short** (p. 3440) instance to be compared

**Returns:**

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

**6.733.2.4 static Short decaf::lang::Short::decode (const std::string & *value*) throw ( exceptions::NumberFormatException ) [static]**

Decodes a **String** (p.3665) into a **Short** (p.3440). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p.3446) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p.3665) is the minus sign. No whitespace characters are permitted in the string.

**Parameters:**

*value* - The string to decode

**Returns:**

a **Short** (p.3440) object containing the decoded value

**Exceptions:**

**NumberFormatException** if the string is not formatted correctly.

**6.733.2.5 virtual double decaf::lang::Short::doubleValue () const [inline, virtual]**

Answers the double value which the receiver represents.

**Returns:**

double the value of the receiver.

Implements **decaf::lang::Number** (p.2836).

**6.733.2.6 bool decaf::lang::Short::equals (const short & *s*) const [inline, virtual]****Returns:**

true if the two **Short** (p.3440) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p.1215).

**6.733.2.7 bool decaf::lang::Short::equals (const Short & *s*) const [inline]****Returns:**

true if the two **Short** (p.3440) Objects have the same value.

**6.733.2.8** `virtual float decaf::lang::Short::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

**Returns:**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.733.2.9** `virtual int decaf::lang::Short::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

**Returns:**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.733.2.10** `virtual long long decaf::lang::Short::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns:**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2836).

**6.733.2.11** `virtual bool decaf::lang::Short::operator< (const short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*s* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< short >** (p. 1215).

**6.733.2.12** `virtual bool decaf::lang::Short::operator< (const Short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*s* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.733.2.13** `virtual bool decaf::lang::Short::operator==(const short & s) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*s* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< short >` (p.1215).

**6.733.2.14** `virtual bool decaf::lang::Short::operator==(const Short & s) const`  
[inline, virtual]

Compares equality between this object and the one passed.

**Parameters:**

*s* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.733.2.15** `static short decaf::lang::Short::parseShort(const std::string & s) throw`  
`( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed decimal short. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort( const std::string, int )` method.

**Parameters:**

*s* - `String` (p.3665) to convert to a short

**Returns:**

the converted short value

**Exceptions:**

*NumberFormatException* if the string is not a short.

**6.733.2.16** `static short decaf::lang::Short::parseShort (const std::string & s, int radix) throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed short in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1103) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:  
 \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1107) or larger than **Character.MAX\_RADIX** (p. 1106). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type short.

**Parameters:**

*s* - the **String** (p. 3665) containing the short representation to be parsed  
*radix* - the radix to be used while parsing *s*

**Returns:**

the short represented by the string argument in the specified radix.

**Exceptions:**

*NumberFormatException* - If **String** (p. 3665) does not contain a parsable short.

**6.733.2.17** `static short decaf::lang::Short::reverseBytes (short value) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

**Parameters:**

*value* - the short whose bytes we are to reverse

**Returns:**

the reversed short.

**6.733.2.18** `virtual short decaf::lang::Short::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns:**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2837).

**6.733.2.19** `static std::string decaf::lang::Short::toString (short value) [static]`

**Returns:**

a string representing the primitive value as Base 10

**6.733.2.20** `std::string decaf::lang::Short::toString () const`**Returns:**

this **Short** (p. 3440) Object as a **String** (p. 3665) Representation

**6.733.2.21** `static Short decaf::lang::Short::valueOf (const std::string & value, int radix) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Short** (p. 3440) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the `parseShort( std::string, int )` method. The result is a **Short** (p. 3440) object that represents the short value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base ( radix )

*radix* - base of the string to parse.

**Returns:**

new **Short** (p. 3440) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a valid short.

**6.733.2.22** `static Short decaf::lang::Short::valueOf (const std::string & value) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Short** (p. 3440) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort( std::string )` method. The result is a **Short** (p. 3440) object that represents the short value specified by the string.

**Parameters:**

*value* - `std::string` to parse as base 10

**Returns:**

new **Short** (p. 3440) Object wrapping the primitive

**Exceptions:**

*NumberFormatException* if the string is not a decimal short.

**6.733.2.23** `static Short decaf::lang::Short::valueOf (short value) [static]`

Returns a **Short** (p. 3440) instance representing the specified short value.

**Parameters:**

*value* - the short to wrap

**Returns:**

the new **Short** (p. 3440) object wrapping value.

**6.733.3 Field Documentation**

**6.733.3.1** `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF`  
[static]

Max Value for this Object's primitive type.

**6.733.3.2** `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

**6.733.3.3** `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Short.h`



## 6.734 decaf::internal::nio::ShortArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/ShortArrayBuffer.h> Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

### Public Member Functions

- **ShortArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **ShortArrayBuffer** (p. 3449) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **ShortArrayBuffer** (short \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **ShortArrayBuffer** (p. 3449) object that wraps the given array.*

- **ShortArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **ShortArrayBuffer** (const ShortArrayBuffer &other)

*Create a **ShortArrayBuffer** (p. 3449) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**ShortArrayBuffer** ()
- virtual short \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the short array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*the array that backs this **Buffer** (p. 928)*

**Exceptions:**

***ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.*

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns:**

*The offset into the backing array where index zero starts.*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this **Buffer** (p. 928) is read only.  
**UnsupportedOperationException** if the underlying store has no array.

- virtual ShortBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only short buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns:**

*The new, read-only short buffer which the caller then owns.*

- virtual ShortBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns:**

*a reference to this **ShortBuffer** (p. 3459).*

**Exceptions:**

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual ShortBuffer \* **duplicate** ()

*Creates a new short buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

*The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns:**

*a new short **Buffer** (p. 928) which the caller owns.*

- virtual short **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns:**

*the short at the current position.*

**Exceptions:**

**BufferUnderflowException** (p. 957) if there no more data to return.

- virtual short **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the short is to be read.

**Returns:**

the short that is located at the given index.

**Exceptions:**

**IndexOutOfBoundsException** if *index* is not smaller than the buffer's limit, or the index is negative.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

- virtual ShortBuffer & **put** (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given shorts into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The shorts value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

**BufferOverflowException** (p. 954) if this buffer's current position is not smaller than its limit.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual ShortBuffer & **put** (int index, short value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given shorts into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The shorts to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

**IndexOutOfBoundsException** if *index* greater than the buffer's limit minus the size of the type being written.

**ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

- virtual ShortBuffer \* **slice** () const

Creates a new **ShortBuffer** (p. 3459) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **ShortBuffer** (p. 3459) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **ShortArrayBuffer** (p. 3449) as Read-Only.

### 6.734.1 Constructor & Destructor Documentation

**6.734.1.1** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (int size, bool readOnly = false) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ShortArrayBuffer** (p. 3449) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

**Parameters:**

*size* The size of the array, this is the limit we read and write to.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

**Exceptions:**

*IllegalArgumentException* if the capacity value is negative.

**6.734.1.2** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * array, int size, int offset, int length, bool readOnly = false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **ShortArrayBuffer** (p. 3449) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

**Parameters:**

*array* The actual array to wrap.

*size* The size of the given array.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if buffer is NULL

*IndexOutOfBoundsException* if offset is greater than array capacity.

**6.734.1.3** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer`  
 (const decaf::lang::Pointer< ByteArrayAdapter > &  
*array*, int *offset*, int *length*, bool *readOnly* = false)  
 throw ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset. The capacity and limit of the new **ShortArrayBuffer** (p. 3449) will be that of the remaining capacity of the passed buffer.

#### Parameters:

*array* The ByteArrayAdapter to wrap.

*offset* The position that is this buffers start position.

*length* The limit of how many bytes into the array this Buffer can write.

*readOnly* Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions:

*NullPointerException* if array is NULL

*IndexOutOfBoundsException* if offset + length is greater than array size.

**6.734.1.4** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer` (const  
 ShortArrayBuffer & *other*)

Create a **ShortArrayBuffer** (p. 3449) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters:

*other* The **ShortArrayBuffer** (p. 3449) this one is to mirror.

**6.734.1.5** `virtual decaf::internal::nio::ShortArrayBuffer::~ShortArrayBuffer` ()  
 [virtual]

## 6.734.2 Member Function Documentation

**6.734.2.1** `virtual short* decaf::internal::nio::ShortArrayBuffer::array` ()  
 throw ( decaf::lang::exceptions::UnsupportedOperationException,  
 decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928)

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3462).

**6.734.2.2** `virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset ()  
throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

***ReadOnlyBufferException*** (p. 3169) if this **Buffer** (p. 928) is read only.

***UnsupportedOperationException*** if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3462).

**6.734.2.3** `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer ()  
const [virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 3462).

#### 6.734.2.4 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact () throw ( decaf::nio::ReadOnlyBufferException ) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

##### Returns:

a reference to this **ShortBuffer** (p. 3459).

##### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3463).

#### 6.734.2.5 virtual ShortBuffer\* decaf::internal::nio::ShortArrayBuffer::duplicate () [virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

##### Returns:

a new short **Buffer** (p. 928) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3463).

#### 6.734.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get (int *index*) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the value at the given index.

##### Parameters:

*index* The index in the **Buffer** (p. 928) where the short is to be read.

##### Returns:

the short that is located at the given index.

**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or the index is negative.

Implements **decaf::nio::ShortBuffer** (p. 3464).

**6.734.2.7** `virtual short decaf::internal::nio::ShortArrayBuffer::get () throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the short at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implements **decaf::nio::ShortBuffer** (p. 3465).

**6.734.2.8** `virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ShortBuffer** (p. 3465).

**6.734.2.9** `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns:**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 931).

**6.734.2.10** `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (int index, short value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given shorts into this buffer at the given index.



**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The shorts to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3465).

**6.734.2.11** `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The shorts value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3466).

**6.734.2.12** `virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 3449) as Read-Only.

**Parameters:**

*value* Boolean value, true if this buffer is to be read-only, false otherwise.

**6.734.2.13** `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice () const` [virtual]

Creates a new **ShortBuffer** (p. 3459) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **ShortBuffer** (p. 3459) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3468).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ShortArrayBuffer.h`

## 6.735 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:.

#include <src/main/decaf/nio/ShortBuffer.h> Inheritance diagram for decaf::nio::ShortBuffer:

### Public Member Functions

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the short array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **ShortBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only short buffer that shares this buffer's content.*
- virtual **ShortBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **ShortBuffer** \* **duplicate** ()=0  
*Creates a new short buffer that shares this buffer's content.*
- virtual short **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual short **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **ShortBuffer** & **get** (std::vector< short > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **ShortBuffer** & **get** (short \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible short array.*
- **ShortBuffer** & **put** (**ShortBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the shorts remaining in the given source buffer into this buffer.*

- **ShortBuffer** & **put** (const short \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers shorts into this buffer from the given source array.*

- **ShortBuffer** & **put** (std::vector< short > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source shorts array into this buffer.*

- virtual **ShortBuffer** & **put** (short value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given shorts into this buffer at the current position, and then increments the position.*

- virtual **ShortBuffer** & **put** (int index, short value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given shorts into this buffer at the given index.*

- virtual **ShortBuffer** \* **slice** () const =0

*Creates a new **ShortBuffer** (p. 3459) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ShortBuffer** &value) const

- virtual bool **equals** (const **ShortBuffer** &value) const

- virtual bool **operator==** (const **ShortBuffer** &value) const

- virtual bool **operator<** (const **ShortBuffer** &value) const

## Static Public Member Functions

- static **ShortBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new Double buffer.*

- static **ShortBuffer** \* **wrap** (short \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **ShortBuffer** (p. 3459).*

- static **ShortBuffer** \* **wrap** (std::vector< short > &buffer)

*Wraps the passed STL short Vector in a **ShortBuffer** (p. 3459).*

## Protected Member Functions

- **ShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ShortBuffer** (p. 3459) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.735.1 Detailed Description

This class defines four categories of operations upon short buffers: . o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.735.2 Constructor & Destructor Documentation

#### 6.735.2.1 decaf::nio::ShortBuffer::ShortBuffer (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **ShortBuffer** (p. 3459) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

##### Parameters:

*capacity* The size and limit of the **Buffer** (p. 928) in doubles

##### Exceptions:

*IllegalArgumentException* if capacity is negative.

#### 6.735.2.2 virtual decaf::nio::ShortBuffer::~~ShortBuffer () [inline, virtual]

### 6.735.3 Member Function Documentation

#### 6.735.3.1 static ShortBuffer\* decaf::nio::ShortBuffer::allocate (int *capacity*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

##### Parameters:

*capacity* The size of the Double buffer in shorts.

##### Returns:

the **ShortBuffer** (p. 3459) that was allocated, caller owns.

**6.735.3.2** `virtual short* decaf::nio::ShortBuffer::array () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the short array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

the array that backs this **Buffer** (p. 928)

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3453).

**6.735.3.3** `virtual int decaf::nio::ShortBuffer::arrayOffset () throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**

The offset into the backing array where index zero starts.

**Exceptions:**

*ReadOnlyBufferException* (p. 3169) if this **Buffer** (p. 928) is read only.

*UnsupportedOperationException* if the underlying store has no array.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3454).

**6.735.3.4** `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns:**

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3454).

### 6.735.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 932) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 932) - 1 is copied to index  $n = \text{limit}()$  (p. 932) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns:

a reference to this **ShortBuffer** (p. 3459).

#### Exceptions:

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3455).

### 6.735.3.6 virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & value) const [virtual]

### 6.735.3.7 virtual ShortBuffer\* decaf::nio::ShortBuffer::duplicate () [pure virtual]

Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns:

a new short **Buffer** (p. 928) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3455).

### 6.735.3.8 virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & value) const [virtual]

### 6.735.3.9 ShortBuffer& decaf::nio::ShortBuffer::get (short \* buffer, int size, int offset, int length) throw (BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if  $\text{length} > \text{remaining}()$  (p. 933), then no bytes are transferred and a **BufferUnderflowException** (p. 957) is thrown.

Otherwise, this method copies length shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

**Parameters:**

*buffer* The pointer to an allocated buffer to fill.  
*size* The size of the buffer provided.  
*offset* The position in the buffer to start filling.  
*length* The amount of data to put in the passed buffer.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length shorts remaining in this buffer  
*NullPointerException* if the passed buffer is null.  
*IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

**6.735.3.10 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*) throw ( BufferUnderflowException )**

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns:**

a reference to this **Buffer** (p. 928).

**Exceptions:**

*BufferUnderflowException* (p. 957) if there are fewer than length shorts remaining in this buffer.

**6.735.3.11 virtual short decaf::nio::ShortBuffer::get (int *index*) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]**

Absolute get method. Reads the value at the given index.

**Parameters:**

*index* The index in the **Buffer** (p. 928) where the short is to be read.

**Returns:**

the short that is located at the given index.



**Exceptions:**

*IndexOutOfBoundsException* if index is not smaller than the buffer's limit, or the index is negative.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3455).

**6.735.3.12 virtual short decaf::nio::ShortBuffer::get () throw ( BufferUnderflowException ) [pure virtual]**

Relative get method. Reads the value at this buffer's current position, and then increments the position.

**Returns:**

the short at the current position.

**Exceptions:**

*BufferUnderflowException* (p. 957) if there no more data to return.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3456).

**6.735.3.13 virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]**

Tells whether or not this buffer is backed by an accessible short array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns:**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3456).

**6.735.3.14 virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & value) const [virtual]**
**6.735.3.15 virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & value) const [virtual]**
**6.735.3.16 virtual ShortBuffer& decaf::nio::ShortBuffer::put (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]**

Writes the given shorts into this buffer at the given index.

**Parameters:**

*index* The position in the **Buffer** (p. 928) to write the data.

*value* The shorts to write.

**Returns:**

a reference to this buffer.

**Exceptions:**

*IndexOutOfBoundsException* if index greater than the buffer's limit minus the size of the type being written.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3456).

**6.735.3.17** `virtual ShortBuffer& decaf::nio::ShortBuffer::put (short value) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

**Parameters:**

*value* The shorts value to be written.

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if this buffer's current position is not smaller than its limit.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 3457).

**6.735.3.18** `ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & buffer) throw ( BufferOverflowException, ReadOnlyBufferException )`

This method transfers the entire content of the given source shorts array into this buffer. This is the same as calling `put( &buffer[0], 0, buffer.size()`.

**Parameters:**

*buffer* The buffer whose contents are copied to this **ShortBuffer** (p. 3459).

**Returns:**

a reference to this buffer.

**Exceptions:**

*BufferOverflowException* (p. 954) if there is insufficient space in this buffer.

*ReadOnlyBufferException* (p. 3169) if this buffer is read-only.

**6.735.3.19** `ShortBuffer& decaf::nio::ShortBuffer::put (const short * buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

This method transfers shorts into this buffer from the given source array. If there are more shorts to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 933), then no shorts are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters:**

- buffer* The array from which shorts are to be read.
- size* The size of the buffer passed.
- offset* The offset within the array of the first char to be read.
- length* The number of shorts to be read from the given array.

**Returns:**

a reference to this buffer.

**Exceptions:**

- BufferOverflowException** (p. 954) if there is insufficient space in this buffer
- ReadOnlyBufferException** (p. 3169) if this buffer is read-only
- NullPointerException** if the passed buffer is null.
- IndexOutOfBoundsException** if the preconditions of size, offset, or length are not met.

### 6.735.3.20 ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & *src*) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the shorts remaining in the given source buffer into this buffer. If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 933), then no shorts are transferred and a **BufferOverflowException** (p. 954) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

**Parameters:**

- src* The buffer to take shorts from an place in this one.

**Returns:**

a reference to this buffer.

**Exceptions:**

- BufferOverflowException** (p. 954) if there is insufficient space in this buffer for the remaining shorts in the source buffer.
- IllegalArgumentException** if the source buffer is this buffer.
- ReadOnlyBufferException** (p. 3169) if this buffer is read-only.

**6.735.3.21** `virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const` [pure virtual]

Creates a new **ShortBuffer** (p.3459) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns:**

the newly create **ShortBuffer** (p.3459) which the caller owns.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p.3457).

**6.735.3.22** `virtual std::string decaf::nio::ShortBuffer::toString () const` [virtual]

**Returns:**

a `std::string` describing this object

**6.735.3.23** `static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & buffer)` [static]

Wraps the passed STL short Vector in a **ShortBuffer** (p.3459). The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

*buffer* The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize( N )`.

**Returns:**

a new **ShortBuffer** (p.3459) that is backed by *buffer*, caller owns.

**6.735.3.24** `static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **ShortBuffer** (p.3459). The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters:**

- array* The array that will back the new buffer.
- size* The size of the passed in array.
- offset* The offset of the subarray to be used.
- length* The length of the subarray to be used.

**Returns:**

- a new **ShortBuffer** (p.3459) that is backed by buffer, caller owns.

**Exceptions:**

- NullPointerException* if the array pointer is NULL.
- IndexOutOfBoundsException* if the preconditions of size, offset, or length are not met.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

## 6.736 activemq::commands::ShutdownInfo Class Reference

#include <src/main/activemq/commands/ShutdownInfo.h> Inheritance diagram for activemq::commands::ShutdownInfo:

### Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ShutdownInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_SHUTDOWNINFO** = 11

## 6.736.1 Constructor & Destructor Documentation

**6.736.1.1** `activemq::commands::ShutdownInfo::ShutdownInfo ()`

**6.736.1.2** `virtual activemq::commands::ShutdownInfo::~~ShutdownInfo ()`  
[virtual]

## 6.736.2 Member Function Documentation

**6.736.2.1** `virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.736.2.2** `virtual void activemq::commands::ShutdownInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.736.2.3** `virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.736.2.4** `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns:

new `DataStructure` (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

**6.736.2.5** `virtual bool activemq::commands::ShutdownInfo::isShutdownInfo () const [inline, virtual]`

**Returns:**

an answer of true to the **isShutdownInfo()** (p. 3472) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 762).

**6.736.2.6** `virtual std::string activemq::commands::ShutdownInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.736.2.7** `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1198).

## 6.736.3 Field Documentation

**6.736.3.1** `const unsigned char activemq::commands::ShutdownInfo::ID__ - SHUTDOWNINFO = 11 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`



## 6.737 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3473).

#include <src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.737.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3473).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.737.2 Constructor & Destructor Documentation

**6.737.2.1** `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.737.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.737.3 Member Function Documentation

**6.737.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.737.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.737.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 794).

**6.737.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 795).

**6.737.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 796).

**6.737.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.737.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ShutdownInfoMarshaller.h**

## 6.738 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3477).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.738.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3477).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.738.2 Constructor & Destructor Documentation

**6.738.2.1** `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.738.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.738.3 Member Function Documentation

**6.738.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.738.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.738.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 801).

**6.738.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 802).

**6.738.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 803).

**6.738.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.738.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**



## 6.739 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3481).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.739.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3481).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.739.2 Constructor & Destructor Documentation

**6.739.2.1** `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.739.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.739.3 Member Function Documentation

**6.739.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.739.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.739.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 780).

**6.739.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 781).

**6.739.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 782).

**6.739.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.739.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**

## 6.740 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3485).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.740.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3485).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.740.2 Constructor & Destructor Documentation

**6.740.2.1** `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.740.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.740.3 Member Function Documentation

**6.740.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.740.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.740.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.740.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.740.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.740.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.740.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**



## 6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3489).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.741.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3489).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.741.2 Constructor & Destructor Documentation

**6.741.2.1** `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.741.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.741.3 Member Function Documentation

**6.741.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.741.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.741.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.741.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.741.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.741.3.6** virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.741.3.7** virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**

## 6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3493).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.742.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3493).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.742.2 Constructor & Destructor Documentation

**6.742.2.1** `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

**6.742.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

## 6.742.3 Member Function Documentation

**6.742.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.742.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.742.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.742.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.742.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.742.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.742.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ShutdownInfoMarshaller.h**



## 6.743 decaf::security::SignatureException Class Reference

#include <src/main/decaf/security/SignatureException.h>Inheritance diagram for decaf::security::SignatureException:

### Public Member Functions

- **SignatureException** () throw ()  
*Default Constructor.*
- **SignatureException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SignatureException** (const **SignatureException** &ex) throw ()  
*Copy Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SignatureException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SignatureException** \* clone () const  
*Clones this exception.*
- virtual ~**SignatureException** () throw ()

### 6.743.1 Constructor & Destructor Documentation

#### 6.743.1.1 decaf::security::SignatureException::SignatureException () throw () [inline]

Default Constructor.

#### 6.743.1.2 decaf::security::SignatureException::SignatureException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.743.1.3 decaf::security::SignatureException::SignatureException (const SignatureException & *ex*) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.743.1.4 decaf::security::SignatureException::SignatureException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.743.1.5 decaf::security::SignatureException::SignatureException (const std::exception \* *cause*) throw () [inline]

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.743.1.6 decaf::security::SignatureException::SignatureException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* name where exception occurs

*lineNumber* line number where the exception occurred.

*msg* message to report

... list of primitives that are formatted into the message

**6.743.1.7** virtual decaf::security::SignatureException::~SignatureException ()  
throw () [inline, virtual]

## 6.743.2 Member Function Documentation

**6.743.2.1** virtual SignatureException\* decaf::security::SignatureException::clone ()  
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1973).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

## 6.744 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 2413) in a human readable format.

#include <src/main/decaf/util/logging/SimpleFormatter.h> Inheritance diagram for decaf::util::logging::SimpleFormatter:

### Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const

*Format the given log record and return the formatted string.*

### 6.744.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 2413) in a human readable format. The summary will typically be 1 or 2 lines.

Since:

1.0

### 6.744.2 Constructor & Destructor Documentation

**6.744.2.1** decaf::util::logging::SimpleFormatter::SimpleFormatter ()

**6.744.2.2** virtual decaf::util::logging::SimpleFormatter::~~SimpleFormatter ()  
[virtual]

### 6.744.3 Member Function Documentation

**6.744.3.1** virtual std::string decaf::util::logging::SimpleFormatter::format (const **LogRecord** & *record*) const [virtual]

Format the given log record and return the formatted string.

**Parameters:**

*record* The Log Record to Format.

Implements **decaf::util::logging::Formatter** (p. 1964).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleFormatter.h**

## 6.745 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

### Public Member Functions

- **SimpleLogger** (const std::string &name)  
*Constructor.*
- virtual **~SimpleLogger** ()  
*Destructor.*
- virtual void **mark** (const std::string &message)  
*Log a Mark Block **Level** (p. 2332) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string &message)  
*Log a Debug **Level** (p. 2332) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string &message)  
*Log a Informational **Level** (p. 2332) Log.*
- virtual void **warn** (const std::string &file, const int line, const std::string &message)  
*Log a Warning **Level** (p. 2332) Log.*
- virtual void **error** (const std::string &file, const int line, const std::string &message)  
*Log a Error **Level** (p. 2332) Log.*
- virtual void **fatal** (const std::string &file, const int line, const std::string &message)  
*Log a Fatal **Level** (p. 2332) Log.*
- virtual void **log** (const std::string &message)  
*No-frills log.*

### 6.745.1 Constructor & Destructor Documentation

#### 6.745.1.1 decaf::util::logging::SimpleLogger::SimpleLogger (const std::string &name)

Constructor.

#### 6.745.1.2 virtual decaf::util::logging::SimpleLogger::~~SimpleLogger () [virtual]

Destructor.

## 6.745.2 Member Function Documentation

**6.745.2.1** `virtual void decaf::util::logging::SimpleLogger::debug (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Debug **Level** (p. 2332) Log.

**6.745.2.2** `virtual void decaf::util::logging::SimpleLogger::error (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Error **Level** (p. 2332) Log.

**6.745.2.3** `virtual void decaf::util::logging::SimpleLogger::fatal (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Fatal **Level** (p. 2332) Log.

**6.745.2.4** `virtual void decaf::util::logging::SimpleLogger::info (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Informational **Level** (p. 2332) Log.

**6.745.2.5** `virtual void decaf::util::logging::SimpleLogger::log (const std::string & message) [virtual]`

No-frills log.

**6.745.2.6** `virtual void decaf::util::logging::SimpleLogger::mark (const std::string & message) [virtual]`

Log a Mark Block **Level** (p. 2332) Log.

**6.745.2.7** `virtual void decaf::util::logging::SimpleLogger::warn (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Warning **Level** (p. 2332) Log.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleLogger.h`

## 6.746 decaf::net::Socket Class Reference

#include <src/main/decaf/net/Socket.h> Inheritance diagram for decaf::net::Socket:

### Public Member Functions

- **Socket** ()  
*Creates an unconnected **Socket** (p. 3503) using the set **SocketImplFactory** (p. 3537) or if none is set than the default **SocketImpl** type is created.*
- **Socket** (**SocketImpl** \*impl)  
*Creates a **Socket** (p. 3503) wrapping the provided **SocketImpl** (p. 3529) instance, this **Socket** (p. 3503) is considered unconnected.*
- **Socket** (const **InetAddress** \*address, int port)  
*Creates a new **Socket** (p. 3503) instance and connects it to the given address and port.*
- **Socket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3503) instance and connects it to the given address and port.*
- **Socket** (const std::string &host, int port)  
*Creates a new **Socket** (p. 3503) instance and connects it to the given host and port.*
- **Socket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3503) instance and connects it to the given host and port.*
- virtual ~**Socket** ()
- virtual void **bind** (const std::string &ipaddress, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Binds this **Socket** (p. 3503) to the given local address and port.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **Socket** (p. 3503).*
- virtual void **connect** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination.*
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination, with a specified timeout value.*
- bool **isConnected** () const  
*Indicates whether or not this socket is connected to an end point.*
- bool **isClosed** () const
- bool **isBound** () const

- **bool isInputShutdown () const**
- **bool isOutputShutdown () const**
- **virtual decaf::io::InputStream \* getInputStream () throw ( decaf::io::IOException )**  
*Gets the InputStream for this socket if its connected.*
- **virtual decaf::io::OutputStream \* getOutputStream () throw ( decaf::io::IOException )**  
*Gets the OutputStream for this socket if it is connected.*
- **int getPort () const**  
*Gets the on the remote host this **Socket** (p. 3503) is connected to.*
- **int getLocalPort () const**  
*Gets the local port the socket is bound to.*
- **std::string getInetAddress () const**  
*Returns the address to which the socket is connected.*
- **std::string getLocalAddress () const**  
*Gets the local address to which the socket is bound.*
- **virtual void shutdownInput () throw ( decaf::io::IOException )**  
*Shuts down the InputStream for this socket essentially marking it as EOF.*
- **virtual void shutdownOutput () throw ( decaf::io::IOException )**  
*Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.*
- **virtual int getSoLinger () const throw ( SocketException )**  
*Gets the linger time for the socket, SO\_LINGER.*
- **virtual void setSoLinger (bool state, int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )**  
*Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.*
- **virtual bool getKeepAlive () const throw ( SocketException )**  
*Gets the keep alive flag for this socket, SO\_KEEPALIVE.*
- **virtual void setKeepAlive (bool keepAlive) throw ( SocketException )**  
*Enables/disables the keep alive flag for this socket, SO\_KEEPALIVE.*
- **virtual int getReceiveBufferSize () const throw ( SocketException )**  
*Gets the receive buffer size for this socket, SO\_RCVBUF.*
- **virtual void setReceiveBufferSize (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )**  
*Sets the receive buffer size for this socket, SO\_RCVBUF.*
- **virtual bool getReuseAddress () const throw ( SocketException )**



*Gets the reuse address flag, SO\_REUSEADDR.*

- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )  
*Sets the reuse address flag, SO\_REUSEADDR.*
- virtual int **getSendBufferSize** () const throw ( SocketException )  
*Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.*
- virtual void **setSendBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.*
- virtual int **getSoTimeout** () const throw ( SocketException )  
*Gets the timeout for socket operations, SO\_TIMEOUT.*
- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the timeout for socket operations, SO\_TIMEOUT.*
- virtual bool **getTcpNoDelay** () const throw ( SocketException )  
*Gets the Status of the TCP\_NODELAY setting for this socket.*
- virtual void **setTcpNoDelay** (bool value) throw ( SocketException )  
*Sets the Status of the TCP\_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3503).*
- virtual int **getTrafficClass** () const throw ( SocketException )  
*Gets the Traffic Class setting for this **Socket** (p. 3503), sometimes referred to as Type of Service setting.*
- virtual void **setTrafficClass** (int value) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Gets the Traffic Class setting for this **Socket** (p. 3503), sometimes referred to as Type of Service setting.*
- virtual bool **getOOBInline** () const throw ( SocketException )  
*Gets the value of the OOBINLINE for this socket.*
- virtual void **setOOBInline** (bool value) throw ( SocketException )  
*Sets the value of the OOBINLINE for this socket, by default this option is disabled.*
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )  
*Sends on byte of urgent data to the **Socket** (p. 3503).*
- virtual std::string **toString** () const

## Static Public Member Functions

- static void **setSocketImplFactory** (**SocketImplFactory** \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )

*Sets the instance of a **SocketImplFactory** (p. 3537) that the **Socket** (p. 3503) class should use when new instances of this class are created.*

## Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (const std::string &address, int port, const **InetAddress** \*localAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )
- void **checkClosed** () const throw ( decaf::io::IOException )
- void **ensureCreated** () const throw ( decaf::io::IOException )

## Protected Attributes

- **SocketImpl** \* impl

## Friends

- class **ServerSocket**

### 6.746.1 Detailed Description

Since:

1.0

### 6.746.2 Constructor & Destructor Documentation

#### 6.746.2.1 decaf::net::Socket::Socket ()

Creates an unconnected **Socket** (p. 3503) using the set **SocketImplFactory** (p. 3537) or if non is set than the default **SocketImpl** type is created.

#### 6.746.2.2 decaf::net::Socket::Socket (**SocketImpl** \* impl)

Creates a **Socket** (p. 3503) wrapping the provided **SocketImpl** (p. 3529) instance, this **Socket** (p. 3503) is considered unconnected. The **Socket** (p. 3503) class takes ownership of this **SocketImpl** (p. 3529) pointer and will delete it when the **Socket** (p. 3503) class is destroyed.

Parameters:

*impl* The **SocketImpl** (p. 3529) instance to wrap.

Exceptions:

*NullPointerException* if the passed **SocketImpl** (p. 3529) is Null.

### 6.746.2.3 decaf::net::Socket::Socket (const InetAddress \* *address*, int *port*)

Creates a new **Socket** (p. 3503) instance and connects it to the given address and port. If there is a **SocketImplFactory** (p. 3537) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3503) implementation is used.

If the host parameter is empty then the loop back address is used.

#### Parameters:

*address* The address to connect to.

*port* The port number to connect to [0...65535]

#### Exceptions:

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*NullPointerException* if the **InetAddress** (p. 2016) instance is NULL.

*IllegalArgumentException* if the port is not in range [0...65535]

### 6.746.2.4 decaf::net::Socket::Socket (const InetAddress \* *address*, int *port*, const InetAddress \* *localAddress*, int *localPort*)

Creates a new **Socket** (p. 3503) instance and connects it to the given address and port. If there is a **SocketImplFactory** (p. 3537) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3503) implementation is used. The **Socket** (p. 3503) will also bind to the local address and port specified.

#### Parameters:

*address* The address to connect to.

*port* The port number to connect to [0...65535]

*localAddress* The IP address on the local machine to bind to.

*localPort* The port on the local machine to bind to.

#### Exceptions:

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*NullPointerException* if the **InetAddress** (p. 2016) instance is NULL.

*IllegalArgumentException* if the port is not in range [0...65535]

### 6.746.2.5 decaf::net::Socket::Socket (const std::string & *host*, int *port*)

Creates a new **Socket** (p. 3503) instance and connects it to the given host and port. If there is a **SocketImplFactory** (p. 3537) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3503) implementation is used.

If the host parameter is empty then the loop back address is used.

**Parameters:**

*host* The host name or IP address to connect to, empty string means loopback.

*port* The port number to connect to [0...65535]

**Exceptions:**

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*IllegalArgumentException* if the port if not in range [0...65535]

#### 6.746.2.6 decaf::net::Socket::Socket (const std::string & *host*, int *port*, const InetAddress \* *localAddress*, int *localPort*)

Creates a new **Socket** (p. 3503) instance and connects it to the given host and port. If there is a **SocketImplFactory** (p. 3537) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3503) implementation is used.

If the host parameter is empty then the loop back address is used.

**Parameters:**

*host* The host name or IP address to connect to, empty string means loopback.

*port* The port number to connect to [0...65535]

*localAddress* The IP address on the local machine to bind to.

*localPort* The port on the local machine to bind to.

**Exceptions:**

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*IllegalArgumentException* if the port if not in range [0...65535]

#### 6.746.2.7 virtual decaf::net::Socket::~~Socket () [virtual]

### 6.746.3 Member Function Documentation

#### 6.746.3.1 void decaf::net::Socket::accepted () [protected]

#### 6.746.3.2 virtual void decaf::net::Socket::bind (const std::string & *ipaddress*, int *port*) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Binds this **Socket** (p. 3503) to the given local address and port. If the **SocketAddress** (p. 3519) value is NULL then the **Socket** (p. 3503) will be bound to an available local address and port.

**Parameters:**

*ipaddress* The local address and port to bind the socket to.

*port* The port on the local machine to bind to.

**Exceptions:**

***IOException*** if an error occurs during the bind operation.

***IllegalArgumentException*** if the **Socket** (p. 3503) can't process the subclass of **SocketAddress** (p. 3519) that has been provided.

**6.746.3.3** `void decaf::net::Socket::checkClosed () const throw ( decaf::io::IOException )` [protected]

**6.746.3.4** `virtual void decaf::net::Socket::close () throw ( decaf::io::IOException )` [virtual]

Closes the **Socket** (p. 3503). Once closed a **Socket** (p. 3503) cannot be connected or otherwise operated upon, a new **Socket** (p. 3503) instance must be created.

**Exceptions:**

***IOException*** if an I/O error occurs while closing the **Socket** (p. 3503).

Implements **decaf::io::Closeable** (p. 1149).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2863).

**6.746.3.5** `virtual void decaf::net::Socket::connect (const std::string & host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Connects to the specified destination, with a specified timeout value. If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3542) is thrown. A timeout value of zero is treated as an infinite timeout.

**Parameters:**

***host*** The host name or IP address of the remote host to connect to.

***port*** The port on the remote host to connect to.

***timeout*** The number of Milliseconds to wait before treating the connection as failed.

**Exceptions:**

***IOException*** Thrown if a failure occurred in the connect.

***SocketTimeoutException*** (p. 3542) if the timeout for connection is exceeded.

***IllegalArgumentException*** if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2863).

**6.746.3.6** `virtual void decaf::net::Socket::connect (const std::string & host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Connects to the specified destination.

**Parameters:**

*host* The host name or IP address of the remote host to connect to.

*port* The port on the remote host to connect to.

**Exceptions:**

*IOException* Thrown if a failure occurred in the connect.

*IllegalArgumentException* if the timeout value is negative or the endpoint is invalid.

**6.746.3.7** `void decaf::net::Socket::ensureCreated () const throw ( decaf::io::IOException )` [protected]

**6.746.3.8** `std::string decaf::net::Socket::getInetAddress () const`

Returns the address to which the socket is connected.

**Returns:**

the remote IP address to which this socket is connected, or null if the socket is not connected.

**6.746.3.9** `virtual decaf::io::InputStream* decaf::net::Socket::getInputStream () throw ( decaf::io::IOException )` [virtual]

Gets the InputStream for this socket if its connected. The pointer returned is the property of the associated **Socket** (p. 3503) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3503).

**Returns:**

The InputStream for this socket.

**Exceptions:**

*IOException* if an error occurs during creation of the InputStream, also if the **Socket** (p. 3503) is not connected or the input has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2864).

**6.746.3.10** `virtual bool decaf::net::Socket::getKeepAlive () const throw ( SocketException )` [virtual]

Gets the keep alive flag for this socket, SO\_KEEPALIVE.

**Returns:**

true if keep alive is enabled for this socket.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.11** `std::string decaf::net::Socket::getLocalAddress () const`

Gets the local address to which the socket is bound.

**Returns:**

the local address to which the socket is bound or `InetAddress.anyLocalAddress()` if the socket is not bound yet.

**6.746.3.12** `int decaf::net::Socket::getLocalPort () const`

Gets the local port the socket is bound to.

**Returns:**

the local port the socket was bound to, or -1 if the socket is not bound.

**6.746.3.13** `virtual bool decaf::net::Socket::getOOBInline () const throw (SocketException) [virtual]`

Gets the value of the OOBINLINE for this socket.

**Returns:**

true if OOBINLINE is enabled, false otherwise.

**Exceptions:**

*SocketException* (p. 3521) if an error is encountered while performing this operation.

**6.746.3.14** `virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream () throw ( decaf::io::IOException) [virtual]`

Gets the OutputStream for this socket if it is connected. The pointer returned is the property of the **Socket** (p. 3503) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3503) will also close the underlying **Socket** (p. 3503).

**Returns:**

the OutputStream for this socket.

**Exceptions:**

*IOException* if an error occurs during the creation of this OutputStream, or if the **Socket** (p. 3503) is closed or the output has been shutdown previously.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2865).

**6.746.3.15** `int decaf::net::Socket::getPort () const`

Gets the on the remote host this **Socket** (p. 3503) is connected to.

**Returns:**

the port on the remote host the socket is connected to, or 0 if not connected.

**6.746.3.16** `virtual int decaf::net::Socket::getReceiveBufferSize () const throw ( SocketException ) [virtual]`

Gets the receive buffer size for this socket, SO\_RCVBUF. This is the buffer used by the underlying platform socket to buffer received data.

**Returns:**

the receive buffer size in bytes.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.17** `virtual bool decaf::net::Socket::getReuseAddress () const throw ( SocketException ) [virtual]`

Gets the reuse address flag, SO\_REUSEADDR.

**Returns:**

True if the address can be reused.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.18** `virtual int decaf::net::Socket::getSendBufferSize () const throw ( SocketException ) [virtual]`

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

**Returns:**

the size in bytes of the send buffer.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.19** `virtual int decaf::net::Socket::getSoLinger () const throw ( SocketException ) [virtual]`

Gets the linger time for the socket, SO\_LINGER. A return value of -1 indicates that the option is disabled.

**Returns:**

The linger time in seconds.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.



**6.746.3.20** `virtual int decaf::net::Socket::getSoTimeout () const throw ( SocketException ) [virtual]`

Gets the timeout for socket operations, SO\_TIMEOUT.

**Returns:**

The timeout in milliseconds for socket operations.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to retrieve the information.

**6.746.3.21** `virtual bool decaf::net::Socket::getTcpNoDelay () const throw ( SocketException ) [virtual]`

Gets the Status of the TCP\_NODELAY setting for this socket.

**Returns:**

true if TCP\_NODELAY is enabled for the socket.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to set the information.

**6.746.3.22** `virtual int decaf::net::Socket::getTrafficClass () const throw ( SocketException ) [virtual]`

Gets the Traffic Class setting for this **Socket** (p. 3503), sometimes referred to as Type of Service setting. This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3503) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

**Returns:**

the bitset result of querying the traffic class setting.

**Exceptions:**

*SocketException* (p. 3521) if an error is encountered while performing this operation.

**6.746.3.23** `void decaf::net::Socket::initSocketImpl (const std::string & address, int port, const InetAddress * localAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [protected]`

**6.746.3.24** `bool decaf::net::Socket::isBound () const [inline]`

**Returns:**

true if this **Socket** (p. 3503) has been bound to a Local address.

**6.746.3.25** `bool decaf::net::Socket::isClosed () const [inline]`**Returns:**

true if the **Socket** (p. 3503) has been closed.

**6.746.3.26** `bool decaf::net::Socket::isConnected () const [inline]`

Indicates whether or not this socket is connected to an end point.

**Returns:**

true if connected, false otherwise.

**6.746.3.27** `bool decaf::net::Socket::isInputShutdown () const [inline]`**Returns:**

true if input on this **Socket** (p. 3503) has been shutdown.

**6.746.3.28** `bool decaf::net::Socket::isOutputShutdown () const [inline]`**Returns:**

true if output on this **Socket** (p. 3503) has been shutdown.

**6.746.3.29** `virtual void decaf::net::Socket::sendUrgentData (int data) throw (decaf::io::IOException) [virtual]`

Sends one byte of urgent data to the **Socket** (p. 3503).

**Parameters:**

*data* The value to write as urgent data, only the lower eight bits are sent.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2866).

**6.746.3.30** `virtual void decaf::net::Socket::setKeepAlive (bool keepAlive) throw (SocketException) [virtual]`

Enables/disables the keep alive flag for this socket, SO\_KEEPALIVE.

**Parameters:**

*keepAlive* If true, enables the flag.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.31** virtual void decaf::net::Socket::setOOBInline (bool *value*) throw ( SocketException ) [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled. If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

**Returns:**

true if OOBINLINE is enabled, false otherwise.

**Exceptions:**

*SocketException* (p. 3521) if an error is encountered while performing this operation.

Reimplemented in decaf::internal::net::ssl::openssl::OpenSSLSocket (p. 2868).

**6.746.3.32** virtual void decaf::net::Socket::setReceiveBufferSize (int *size*) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Sets the receive buffer size for this socket, SO\_RCVBUF.

**Parameters:**

*size* Number of bytes to set the receive buffer to.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

*IllegalArgumentException* if the value is zero or negative.

**6.746.3.33** virtual void decaf::net::Socket::setReuseAddress (bool *reuse*) throw ( SocketException ) [virtual]

Sets the reuse address flag, SO\_REUSEADDR.

**Parameters:**

*reuse* If true, sets the flag.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

**6.746.3.34** virtual void decaf::net::Socket::setSendBufferSize (int *size*) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

**Parameters:**

*size* The number of bytes to set the send buffer to, must be larger than zero.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

*IllegalArgumentException* if the value is zero or negative.

**6.746.3.35**    `static void decaf::net::Socket::setSocketImplFactory (SocketImplFactory * factory) throw ( decaf::io::IOException, decaf::net::SocketException )`  
[static]

Sets the instance of a **SocketImplFactory** (p. 3537) that the **Socket** (p. 3503) class should use when new instances of this class are created. This method is only allowed to be used once during the lifetime of the application.

**Parameters:**

*factory* The instance of a **SocketImplFactory** (p. 3537) to use when new **Socket** (p. 3503) objects are created.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

*SocketException* (p. 3521) if this method has already been called with a valid factory.

**6.746.3.36**    `virtual void decaf::net::Socket::setSoLinger (bool state, int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.

**Parameters:**

*state* The state of SO\_LINGER, true is on.

*timeout* The linger time in seconds, must be non-negative.

**Exceptions:**

*SocketException* (p. 3521) if the operation fails.

*IllegalArgumentException* if state is true and timeout is negative.

**6.746.3.37**    `virtual void decaf::net::Socket::setSoTimeout (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Sets the timeout for socket operations, SO\_TIMEOUT. A value of zero indicates that timeout is infinite for operations on this socket.

**Parameters:**

*timeout* The timeout in milliseconds for socket operations.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to set the information.

*IllegalArgumentException* if the timeout value is negative.

**6.746.3.38** `virtual void decaf::net::Socket::setTcpNoDelay (bool value) throw (SocketException )` [virtual]

Sets the Status of the TCP\_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3503).

**Parameters:**

*value* The setting for the socket's TCP\_NODELAY option, true to enable.

**Exceptions:**

*SocketException* (p. 3521) Thrown if unable to set the information.

**6.746.3.39** `virtual void decaf::net::Socket::setTrafficClass (int value) throw (SocketException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Gets the Traffic Class setting for this **Socket** (p. 3503), sometimes referred to as Type of Service setting. This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3503) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

**Parameters:**

*value* The integer value representing the traffic class setting bitset.

**Exceptions:**

*SocketException* (p. 3521) if an error is encountered while performing this operation.

*IllegalArgumentException* if the value is not in the range [0..255].

**6.746.3.40** `virtual void decaf::net::Socket::shutdownInput () throw (decaf::io::IOException )` [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF. The stream returns EOF for any calls to read after this method has been called.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2869).

**6.746.3.41** `virtual void decaf::net::Socket::shutdownOutput () throw (decaf::io::IOException ) [virtual]`

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to `OutputStream::write` will throw an `IOException`.

**Exceptions:**

***IOException*** if an I/O error occurs while performing this operation.

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2869).

**6.746.3.42** `virtual std::string decaf::net::Socket::toString () const [virtual]`

**Returns:**

a string representing this `Socket` (p. 3503).

## 6.746.4 Friends And Related Function Documentation

**6.746.4.1** `friend class ServerSocket [friend]`

## 6.746.5 Field Documentation

**6.746.5.1** `SocketImpl* decaf::net::Socket::impl [mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

## 6.747 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 3503) addresses.

`#include <src/main/decaf/net/SocketAddress.h>`Inheritance diagram for decaf::net::SocketAddress:

### Public Member Functions

- virtual `~SocketAddress ()`

#### 6.747.1 Detailed Description

Base class for protocol specific **Socket** (p. 3503) addresses. These classes provide an immutable address object that is used by the **Socket** (p. 3503) classes.

**Since:**

1.0

#### 6.747.2 Constructor & Destructor Documentation

##### 6.747.2.1 virtual decaf::net::SocketAddress::~~SocketAddress () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketAddress.h`

## 6.748 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

### Static Public Member Functions

- static int **getErrorCode** ()  
*Gets the last error appropriate for the platform.*
- static std::string **getErrorString** ()  
*Gets the string description for the last error.*

### 6.748.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

### 6.748.2 Member Function Documentation

#### 6.748.2.1 static int decaf::net::SocketError::getErrorCode () [static]

Gets the last error appropriate for the platform.

#### 6.748.2.2 static std::string decaf::net::SocketError::getErrorString () [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketError.h**



## 6.749 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

#include <src/main/decaf/net/SocketException.h> Inheritance diagram for decaf::net::SocketException:

### Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const lang::Exception &ex) throw ()
- **SocketException** (const SocketException &ex) throw ()
- **SocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SocketException** \* clone () const  
*Clones this exception.*
- virtual ~**SocketException** () throw ()

### 6.749.1 Detailed Description

Exception for errors when manipulating sockets.

### 6.749.2 Constructor & Destructor Documentation

- 6.749.2.1 decaf::net::SocketException::SocketException () throw () [inline]
- 6.749.2.2 decaf::net::SocketException::SocketException (const lang::Exception &ex) throw () [inline]
- 6.749.2.3 decaf::net::SocketException::SocketException (const SocketException &ex) throw () [inline]
- 6.749.2.4 decaf::net::SocketException::SocketException (const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs  
*lineNumber* The line number where the exception occurred.  
*cause* The exception that was the cause for this one to be thrown.  
*msg* The message to report  
... list of primitives that are formatted into the message

**6.749.2.5** `decaf::net::SocketException::SocketException (const std::exception *  
cause) throw () [inline]`

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.749.2.6** `decaf::net::SocketException::SocketException (const char * file, const int  
lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs  
*lineNumber* The line number where the exception occurred.  
*msg* The message to report  
... list of primitives that are formatted into the message

**6.749.2.7** `virtual decaf::net::SocketException::~SocketException () throw ()  
[inline, virtual]`

**6.749.3 Member Function Documentation**

**6.749.3.1** `virtual SocketException* decaf::net::SocketException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2144).

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketException` (p. 2873), `decaf::net::BindException` (p. 838), `decaf::net::ConnectException`

(p. 1261), **decaf::net::NoRouteToHostException** (p. 2822), and **decaf::net::PortUnreachableException** (p. 2975).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketException.h`

## 6.750 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 3524) is used to create **Socket** (p. 3503) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

#include <src/main/decaf/net/SocketFactory.h> Inheritance diagram for decaf::net::SocketFactory:

### Public Member Functions

- virtual **~SocketFactory** ()
- virtual **Socket \* createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3503) object.*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port, const **InetAddress** \*ifAddress, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*
- virtual **Socket \* createSocket** (const std::string &name, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*
- virtual **Socket \* createSocket** (const std::string &name, int port, const **InetAddress** \*ifAddress, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).*

### Static Public Member Functions

- static **SocketFactory \* getDefault** ()  
*Returns an pointer to the default **SocketFactory** (p. 3524) for this Application, there is only one default **SocketFactory** (p. 3524) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3524) class and in not to be deleted by the caller.*

### Protected Member Functions

- **SocketFactory** ()

## 6.750.1 Detailed Description

The **SocketFactory** (p. 3524) is used to create **Socket** (p. 3503) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

See also:

`decaf.net.Socket` (p. 3503)

Since:

1.0

## 6.750.2 Constructor & Destructor Documentation

**6.750.2.1** `decaf::net::SocketFactory::SocketFactory ()` [protected]

**6.750.2.2** `virtual decaf::net::SocketFactory::~~SocketFactory ()` [virtual]

## 6.750.3 Member Function Documentation

**6.750.3.1** `virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & name, int port, const InetAddress * ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )` [pure virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

Parameters:

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

Returns:

a new **Socket** (p. 3503) object, caller must free this object when done.

Exceptions:

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implemented in `decaf::internal::net::DefaultSocketFactory` (p. 1688), `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p. 1700), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2878).

**6.750.3.2** `virtual Socket* decaf::net::SocketFactory::createSocket (const std::string & name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )` [pure virtual]

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host name or IP address to connect the socket to.

*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1688), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1700), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2878).

**6.750.3.3** `virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * host, int port, const InetAddress * ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [pure virtual]`

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524). The **Socket** (p. 3503) will be bound to the specified local address and port.

**Parameters:**

*host* The host to connect the socket to.

*port* The port on the remote host to connect to.

*ifAddress* The address on the local machine to bind the **Socket** (p. 3503) to.

*localPort* The local port to bind the **Socket** (p. 3503) to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.

*UnknownHostException* (p. 3907) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1689), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1701), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2879).

**6.750.3.4** `virtual Socket* decaf::net::SocketFactory::createSocket (const InetAddress * host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [pure virtual]`

Creates a new **Socket** (p. 3503) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3524).

**Parameters:**

*host* The host to connect the socket to.  
*port* The port on the remote host to connect to.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if an I/O error occurs while creating the **Socket** (p. 3503) object.  
*UnknownHostException* (p. 3907) if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1689), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1701), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2879).

### 6.750.3.5 virtual Socket\* decaf::net::SocketFactory::createSocket () throw ( decaf::io::IOException ) [virtual]

Creates an unconnected **Socket** (p. 3503) object.

**Returns:**

a new **Socket** (p. 3503) object, caller must free this object when done.

**Exceptions:**

*IOException* if the **Socket** (p. 3503) cannot be created.

Reimplemented in **decaf::internal::net::DefaultSocketFactory** (p. 1690), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1702), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2880).

### 6.750.3.6 static SocketFactory\* decaf::net::SocketFactory::getDefault () [static]

Returns an pointer to the default **SocketFactory** (p. 3524) for this Application, there is only one default **SocketFactory** (p. 3524) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3524) class and in not to be deleted by the caller.

**Returns:**

pointer to the applications default **SocketFactory** (p. 3524).

**Exceptions:**

*SocketException* (p. 3521) if an error occurs while getting the default instance.

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 3572).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**

## 6.751 decaf::internal::net::SocketFileDescriptor Class Reference

File Descriptor type used internally by Decaf Socket objects.

#include <src/main/decaf/internal/net/SocketFileDescriptor.h>Inheritance diagram for decaf::internal::net::SocketFileDescriptor:

### Public Member Functions

- **SocketFileDescriptor** (long value)
- virtual **~SocketFileDescriptor** ()
- long **getValue** () const

*Gets the OS Level FileDescriptor.*

### 6.751.1 Detailed Description

File Descriptor type used internally by Decaf Socket objects.

**Since:**

1.0

### 6.751.2 Constructor & Destructor Documentation

**6.751.2.1** decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor (long *value*)

**6.751.2.2** virtual decaf::internal::net::SocketFileDescriptor::~~SocketFileDescriptor () [virtual]

### 6.751.3 Member Function Documentation

**6.751.3.1** long decaf::internal::net::SocketFileDescriptor::getValue () const

Gets the OS Level FileDescriptor.

**Returns:**

a FileDescriptor value.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**SocketFileDescriptor.h**



## 6.752 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 3503) implementations.

#include <src/main/decaf/net/SocketImpl.h> Inheritance diagram for decaf::net::SocketImpl:

### Public Member Functions

- **SocketImpl** ()
- virtual **~SocketImpl** ()
- virtual void **create** ()=0 throw ( decaf::io::IOException )  
*Creates the underlying platform **Socket** (p. 3503) data structures which allows for **Socket** (p. 3503) options to be applied.*
- virtual void **accept** (**SocketImpl** \*socket)=0 throw ( decaf::io::IOException, decaf::net::SocketException, decaf::net::SocketTimeoutException )  
*Accepts a new connection on the given **Socket** (p. 3503).*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout)=0 throw ( decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects this socket to the given host and port.*
- virtual void **bind** (const std::string &ipaddress, int **port**)=0 throw ( decaf::io::IOException )  
*Binds this **Socket** (p. 3503) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0 throw ( decaf::io::IOException )  
*Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.*
- virtual **decaf::io::InputStream** \* **getInputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the **InputStream** linked to this **Socket** (p. 3503).*
- virtual **decaf::io::OutputStream** \* **getOutputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the **OutputStream** linked to this **Socket** (p. 3503).*
- virtual int **available** ()=0 throw ( decaf::io::IOException )  
*Gets the number of bytes that can be read from the **Socket** (p. 3503) without blocking.*
- virtual void **close** ()=0 throw ( decaf::io::IOException )  
*Closes the socket, terminating any blocked reads or writes.*
- virtual void **shutdownInput** ()=0 throw ( decaf::io::IOException )  
*Places the input stream for this socket at "end of stream".*

- virtual void **shutdownOutput** ()=0 throw ( decaf::io::IOException )  
*Disables the output stream for this socket.*
- virtual int **getOption** (int option) const =0 throw ( decaf::io::IOException )  
*Gets the specified **Socket** (p. 3503) option.*
- virtual void **setOption** (int option, int value)=0 throw ( decaf::io::IOException )  
*Sets the specified option on the **Socket** (p. 3503) if supported.*
- int **getPort** () const  
*Gets the port that this socket has been assigned.*
- int **getLocalPort** () const  
*Gets the value of this SocketImpl's local port field.*
- std::string **getInetAddress** () const  
*Gets the value of this SocketImpl's address field.*
- const decaf::io::FileDescriptor \* **getFileDescriptor** () const  
*Gets the FileDescriptor for this **Socket** (p. 3503), the Object is owned by this **Socket** (p. 3503) and should not be deleted by the caller.*
- virtual std::string **getLocalAddress** () const =0  
*Gets the value of the local Inet address the **Socket** (p. 3503) is bound to if bound, otherwise return the **InetAddress** (p. 2016) ANY value "0.0.0.0".*
- std::string **toString** () const  
*Returns a string containing the address and port of this **Socket** (p. 3503) instance.*
- virtual bool **supportsUrgentData** () const
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )  
*Sends on byte of urgent data to the **Socket** (p. 3503).*

## Protected Attributes

- int **port**  
*The remote port that this **Socket** (p. 3503) is connected to.*
- int **localPort**  
*The port on the Local Machine that this **Socket** (p. 3503) is Bound to.*
- std::string **address**  
*The Remote Address that the **Socket** (p. 3503) is connected to.*
- io::FileDescriptor \* **fd**  
*The File Descriptor for this **Socket** (p. 3503).*

### 6.752.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 3503) implementations.

Since:

1.0

### 6.752.2 Constructor & Destructor Documentation

**6.752.2.1** decaf::net::SocketImpl::SocketImpl ()

**6.752.2.2** virtual decaf::net::SocketImpl::~~SocketImpl () [virtual]

### 6.752.3 Member Function Documentation

**6.752.3.1** virtual void decaf::net::SocketImpl::accept (SocketImpl \* *socket*)  
throw ( decaf::io::IOException, decaf::net::SocketException,  
decaf::net::SocketTimeoutException ) [pure virtual]

Accepts a new connection on the given **Socket** (p. 3503).

Parameters:

*socket* The accepted connection.

Exceptions:

*IOException* if an I/O error occurs while attempting this operation.

*SocketException* (p. 3521) if an error occurs while performing an Accept on the socket.

*SocketTimeoutException* (p. 3542) if the accept call times out due to SO\_TIMEOUT being set.

**6.752.3.2** virtual int decaf::net::SocketImpl::available () throw (  
decaf::io::IOException ) [pure virtual]

Gets the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

Returns:

the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

Exceptions:

*IOException* if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3741).

**6.752.3.3** virtual void decaf::net::SocketImpl::bind (const std::string & *ipaddress*,  
int *port*) throw ( decaf::io::IOException ) [pure virtual]

Binds this **Socket** (p. 3503) instance to the local ip address and port number given.

**Parameters:**

*ipaddress* The address of local ip to bind to.  
*port* The port number on the host to bind to.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3741).

**6.752.3.4 virtual void decaf::net::SocketImpl::close () throw ( decaf::io::IOException ) [pure virtual]**

Closes the socket, terminating any blocked reads or writes.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3742).

**6.752.3.5 virtual void decaf::net::SocketImpl::connect (const std::string & hostname, int port, int timeout) throw ( decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]**

Connects this socket to the given host and port.

**Parameters:**

*hostname* The name of the host to connect to, or IP address.  
*port* The port number on the host to connect to.  
*timeout* Time in milliseconds to wait for a connection, 0 indicates forever.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.  
*SocketTimeoutException* (p. 3542) if the connect call times out due to timeout being set.  
*IllegalArgumentException* if a parameter has an illegal value.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3742).

**6.752.3.6 virtual void decaf::net::SocketImpl::create () throw ( decaf::io::IOException ) [pure virtual]**

Creates the underlying platform **Socket** (p. 3503) data structures which allows for **Socket** (p. 3503) options to be applied. The created socket is in an unconnected state.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3742).

**6.752.3.7** `const decaf::io::FileDescriptor* decaf::net::SocketImpl::getFileDescriptor () const [inline]`

Gets the FileDescriptor for this **Socket** (p. 3503), the Object is owned by this **Socket** (p. 3503) and should not be deleted by the caller.

**Returns:**

a pointer to this Socket's FileDescriptor object.

**6.752.3.8** `std::string decaf::net::SocketImpl::getInetAddress () const [inline]`

Gets the value of this SocketImpl's address field.

**Returns:**

the value of the address field.

**6.752.3.9** `virtual decaf::io::InputStream* decaf::net::SocketImpl::getInputStream () throw ( decaf::io::IOException ) [pure virtual]`

Gets the InputStream linked to this **Socket** (p. 3503).

**Returns:**

an InputStream pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3743).

**6.752.3.10** `virtual std::string decaf::net::SocketImpl::getLocalAddress () const [pure virtual]`

Gets the value of the local Inet address the **Socket** (p. 3503) is bound to if bound, otherwise return the **InetAddress** (p. 2016) ANY value "0.0.0.0".

**Returns:**

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3743).

**6.752.3.11** `int decaf::net::SocketImpl::getLocalPort () const [inline]`

Gets the value of this SocketImpl's local port field.

**Returns:**

the value of localPort.

**6.752.3.12** `virtual int decaf::net::SocketImpl::getOption (int option) const throw ( decaf::io::IOException )` [pure virtual]

Gets the specified **Socket** (p. 3503) option.

**Parameters:**

*option* The **Socket** (p. 3503) options whose value is to be retrieved.

**Returns:**

the value of the given socket option.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3743).

**6.752.3.13** `virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream () throw ( decaf::io::IOException )` [pure virtual]

Gets the OutputStream linked to this **Socket** (p. 3503).

**Returns:**

an OutputStream pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3743).

**6.752.3.14** `int decaf::net::SocketImpl::getPort () const` [inline]

Gets the port that this socket has been assigned.

**Returns:**

the Socket's port number.

**6.752.3.15** `virtual void decaf::net::SocketImpl::listen (int backlog) throw ( decaf::io::IOException )` [pure virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument. If a connection indication arrives when the queue is full, the connection is refused.

**Parameters:**

*backlog* The maximum length of the connection queue.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3744).

**6.752.3.16** virtual void decaf::net::SocketImpl::sendUrgentData (int *data*) throw ( decaf::io::IOException ) [virtual]

Sends on byte of urgent data to the **Socket** (p. 3503).

**Parameters:**

*data* The value to write as urgent data, only the lower eight bits are sent.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

**6.752.3.17** virtual void decaf::net::SocketImpl::setOption (int *option*, int *value*) throw ( decaf::io::IOException ) [pure virtual]

Sets the specified option on the **Socket** (p. 3503) if supported.

**Parameters:**

*option* The **Socket** (p. 3503) option to set.

*value* The value of the socket option to apply to the socket.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3745).

**6.752.3.18** virtual void decaf::net::SocketImpl::shutdownInput () throw ( decaf::io::IOException ) [pure virtual]

Places the input stream for this socket at "end of stream". Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3535) on the socket, the stream will return EOF.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3745).

**6.752.3.19** `virtual void decaf::net::SocketImpl::shutdownOutput () throw ( decaf::io::IOException ) [pure virtual]`

Disables the output stream for this socket. For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking `shutdownOutput()` (p. 3536) on the socket, the stream will throw an `IOException`.

**Exceptions:**

***IOException*** if an I/O error occurs while attempting this operation.

Implemented in `decaf::internal::net::tcp::TcpSocket` (p. 3746).

**6.752.3.20** `virtual bool decaf::net::SocketImpl::supportsUrgentData () const [inline, virtual]`

**Returns:**

true if this **SocketImpl** (p. 3529) supports sending Urgent Data. The default implementation always returns false.

**6.752.3.21** `std::string decaf::net::SocketImpl::toString () const`

Returns a string containing the address and port of this **Socket** (p. 3503) instance.

**Returns:**

a string containing the address and port of this socket.

## 6.752.4 Field Documentation

**6.752.4.1** `std::string decaf::net::SocketImpl::address [protected]`

The Remote Address that the **Socket** (p. 3503) is connected to.

**6.752.4.2** `io::FileDescriptor* decaf::net::SocketImpl::fd [protected]`

The File Descriptor for this **Socket** (p. 3503).

**6.752.4.3** `int decaf::net::SocketImpl::localPort [protected]`

The port on the Local Machine that this **Socket** (p. 3503) is Bound to.

**6.752.4.4** `int decaf::net::SocketImpl::port [protected]`

The remote port that this **Socket** (p. 3503) is connected to.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImpl.h`



## 6.753 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates SocketImpl objects.

```
#include <src/main/decaf/net/SocketImplFactory.h>
```

### Public Member Functions

- virtual `~SocketImplFactory ()`
- virtual `SocketImpl * createSocketImpl ()=0`

*Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3529).*

### 6.753.1 Detailed Description

Factory class interface for a Factory that creates SocketImpl objects. These factories can be used to create various types of Sockets, e.g. Streaming, Multicast, SSL, or platform specific variations of these types.

See also:

`decaf::net::Socket` (p. 3503)  
`decaf::net::ServerSocket` (p. 3352)

Since:

1.0

### 6.753.2 Constructor & Destructor Documentation

**6.753.2.1** virtual `decaf::net::SocketImplFactory::~~SocketImplFactory ()` [inline, virtual]

### 6.753.3 Member Function Documentation

**6.753.3.1** virtual `SocketImpl* decaf::net::SocketImplFactory::createSocketImpl ()` [pure virtual]

Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3529).

Returns:

new **SocketImpl** (p. 3529) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImplFactory.h`

## 6.754 decaf::net::SocketOptions Class Reference

#include <src/main/decaf/net/SocketOptions.h> Inheritance diagram for decaf::net::SocketOptions:

### Public Member Functions

- virtual `~SocketOptions()`

### Static Public Attributes

- static const int **SOCKET\_OPTION\_TCP\_NODELAY**  
*Disable Nagle's algorithm for this connection.*
- static const int **SOCKET\_OPTION\_BINDADDR**  
*Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).*
- static const int **SOCKET\_OPTION\_REUSEADDR**  
*Sets SO\_REUSEADDR for a socket.*
- static const int **SOCKET\_OPTION\_BROADCAST**  
*Sets SO\_BROADCAST for a socket.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF**  
*Set which outgoing interface on which to send multicast packets.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF2**  
*Same as above.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_LOOP**  
*This option enables or disables local loopback of multicast datagrams.*
- static const int **SOCKET\_OPTION\_IP\_TOS**  
*This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.*
- static const int **SOCKET\_OPTION\_LINGER**  
*Specify a linger-on-close timeout.*
- static const int **SOCKET\_OPTION\_TIMEOUT**  
*Set a timeout on blocking **Socket** (p. 3503) operations.*
- static const int **SOCKET\_OPTION\_SNDBUF**  
*Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.*
- static const int **SOCKET\_OPTION\_RCVBUF**

*Set a hint the size of the underlying buffers used by the platform for incoming network I/O.*

- static const int **SOCKET\_OPTION\_KEEPAKALIVE**

*When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.*

- static const int **SOCKET\_OPTION\_OOBINLINE**

*When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.*

## 6.754.1 Detailed Description

Since:

1.0

## 6.754.2 Constructor & Destructor Documentation

**6.754.2.1** virtual decaf::net::SocketOptions::~SocketOptions () [virtual]

## 6.754.3 Field Documentation

**6.754.3.1** const int decaf::net::SocketOptions::SOCKET\_OPTION\_BINDADDR [static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed). The default local address of a socket is INADDR\_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 3352) or **DatagramSocket**), or to specify its return address to the peer (for a **Socket** (p. 3503) or **DatagramSocket**). The parameter of this option is an **InetAddress** (p. 2016).

**6.754.3.2** const int decaf::net::SocketOptions::SOCKET\_OPTION\_BROADCAST [static]

Sets SO\_BROADCAST for a socket. This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for **DatagramSockets**.

**6.754.3.3** const int decaf::net::SocketOptions::SOCKET\_OPTION\_IP\_MULTICAST\_IF [static]

Set which outgoing interface on which to send multicast packets. Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 2016).

Valid for Multicast: **DatagramSocketImpl**.

**6.754.3.4** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_IF2` [static]

Same as above. This option is introduced so that the behaviour with `IP_MULTICAST_IF` will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

**6.754.3.5** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_MULTICAST_LOOP` [static]

This option enables or disables local loopback of multicast datagrams. This option is enabled by default for Multicast Sockets.

**6.754.3.6** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS` [static]

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

**6.754.3.7** `const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPAIVE` [static]

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer. This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 3529)

**6.754.3.8** `const int decaf::net::SocketOptions::SOCKET_OPTION_LINGER` [static]

Specify a linger-on-close timeout. This option disables/enables immediate return from a `close()` of a TCP **Socket** (p. 3503). Enabling this option with a non-zero Integer timeout means that a `close()` will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 3529)

**6.754.3.9** `const int decaf::net::SocketOptions::SOCKET_OPTION_OOBLINE` [static]

When the OOBLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream. When the option is disabled (which is the default) urgent data is silently discarded.

**6.754.3.10** `const int decaf::net::SocketOptions::SOCKET_OPTION_RCVBUF`  
[static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O. When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 3529), **DatagramSocketImpl**.

**6.754.3.11** `const int decaf::net::SocketOptions::SOCKET_OPTION_SO_REUSEADDR` [static]

Sets SO\_REUSEADDR for a socket. This is used only for MulticastSockets in **decaf** (p. 143), and it is set by default for MulticastSockets.

**6.754.3.12** `const int decaf::net::SocketOptions::SOCKET_OPTION_SNDBUF`  
[static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O. When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 3529), **DatagramSocketImpl**.

**6.754.3.13** `const int decaf::net::SocketOptions::SOCKET_OPTION_TCP_NODELAY` [static]

Disable Nagle's algorithm for this connection. Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

**6.754.3.14** `const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT`  
[static]

Set a timeout on blocking **Socket** (p. 3503) operations. The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOptions.h`

## 6.755 decaf::net::SocketTimeoutException Class Reference

#include <src/main/decaf/net/SocketTimeoutException.h> Inheritance diagram for decaf::net::SocketTimeoutException:

### Public Member Functions

- **SocketTimeoutException** () throw ()  
*Default Constructor.*
- **SocketTimeoutException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()  
*Copy Constructor.*
- **SocketTimeoutException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketTimeoutException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SocketTimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SocketTimeoutException** \* **clone** () const  
*Clones this exception.*
- virtual ~**SocketTimeoutException** () throw ()

### 6.755.1 Constructor & Destructor Documentation

#### 6.755.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException () throw () [inline]

Default Constructor.

#### 6.755.1.2 decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.755.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.755.1.4 decaf::net::SocketTimeoutException::SocketTimeoutException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.755.1.5 decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.755.1.6 decaf::net::SocketTimeoutException::SocketTimeoutException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.755.1.7** `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException  
() throw () [inline, virtual]`

## **6.755.2 Member Function Documentation**

**6.755.2.1** `virtual SocketTimeoutException* de-  
caf::net::SocketTimeoutException::clone () const [inline,  
virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p. 2129).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketTimeoutException.h`



## 6.756 decaf::net::ssl::SSLContext Class Reference

Represents on implementation of the Secure **Socket** (p. 3503) Layer for streaming based sockets.

```
#include <src/main/decaf/net/ssl/SSLContext.h>
```

### Public Member Functions

- **SSLContext** (**SSLContextSpi** \*contextImpl)
- virtual **~SSLContext** ()
- **SocketFactory** \* **getSocketFactory** ()  
*Returns an **SocketFactory** (p. 3524) instance for use with this Context, the **SocketFactory** (p. 3524) is owned by the Context and should not be deleted by the caller.*
- **ServerSocketFactory** \* **getServerSocketFactory** ()  
*Returns an **ServerSocketFactory** (p. 3361) instance for use with this Context, the **ServerSocketFactory** (p. 3361) is owned by the Context and should not be deleted by the caller.*
- **SSLParameters** \* **getDefaultSSLParameters** ()
- **SSLParameters** \* **getSupportedSSLParameters** ()

### Static Public Member Functions

- static **SSLContext** \* **getDefault** ()  
*Gets the Default **SSLContext** (p. 3545).*
- static void **setDefault** (**SSLContext** \*context)  
*Sets the default **SSLContext** (p. 3545) to be returned from future calls to **getDefault**.*

### 6.756.1 Detailed Description

Represents on implementation of the Secure **Socket** (p. 3503) Layer for streaming based sockets. This class servers a a source of factories to be used to create new SSL **Socket** (p. 3503) instances.

Since:

1.0

### 6.756.2 Constructor & Destructor Documentation

**6.756.2.1** decaf::net::ssl::SSLContext::SSLContext (**SSLContextSpi** \* contextImpl)

**6.756.2.2** virtual decaf::net::ssl::SSLContext::~~SSLContext () [virtual]

### 6.756.3 Member Function Documentation

**6.756.3.1** static **SSLContext**\* decaf::net::ssl::SSLContext::getDefault () [static]

Gets the Default **SSLContext** (p. 3545). The default instance of the **SSLContext** (p. 3545) should be immediately usable without any need for the client to initialize this context.

**Returns:**

a pointer to the Default **SSLContext** (p. 3545) instance.

**6.756.3.2 SSLParameters\* decaf::net::ssl::SSLContext::getDefaultSSLParameters ()****Returns:**

a new instance of an **SSLParameters** (p. 3551) object containing the default set of settings for this **SSLContext** (p. 3545).

**Exceptions:**

*UnsupportedOperationException* if the parameters cannot be retrieved.

**6.756.3.3 ServerSocketFactory\* decaf::net::ssl::SSLContext::getServerSocketFactory ()**

Returns an **ServerSocketFactory** (p. 3361) instance for use with this Context, the **ServerSocketFactory** (p. 3361) is owned by the Context and should not be deleted by the caller.

**Returns:**

a pointer to this **SSLContext**'s **ServerSocketFactory** (p. 3361) for creating **SSLServerSocket** (p. 3554) objects.

**Exceptions:**

*IllegalStateException* if the **SSLContextSpi** (p. 3548) requires initialization but it has not yet been initialized.

**6.756.3.4 SocketFactory\* decaf::net::ssl::SSLContext::getSocketFactory ()**

Returns an **SocketFactory** (p. 3524) instance for use with this Context, the **SocketFactory** (p. 3524) is owned by the Context and should not be deleted by the caller.

**Returns:**

a pointer to this **SSLContext**'s **SocketFactory** (p. 3524) for creating **SSLSocket** (p. 3563) objects.

**Exceptions:**

*IllegalStateException* if the **SSLContextSpi** (p. 3548) requires initialization but it has not yet been initialized.

**6.756.3.5 SSLParameters\* decaf::net::ssl::SSLContext::getSupportedSSLParameters ()****Returns:**

a new instance of an **SSLParameters** (p. 3551) object containing the complete set of settings for this **SSLContext** (p. 3545).

**Exceptions:**

*UnsupportedOperationException* if the parameters cannot be retrieved.

**6.756.3.6 static void decaf::net::ssl::SSLContext::setDefault (SSLContext \* *context*)**  
[static]

Sets the default **SSLContext** (p. 3545) to be returned from future calls to getDefault. The set **SSLContext** (p. 3545) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

**Exceptions:**

*NullPointerException* if the context passed is NULL.

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLContext.h**

## 6.757 decaf::net::ssl::SSLContextSpi Class Reference

Defines the interface that should be provided by an **SSLContext** (p. 3545) provider.

#include <src/main/decaf/net/ssl/SSLContextSpi.h> Inheritance diagram for decaf::net::ssl::SSLContextSpi:

### Public Member Functions

- virtual **~SSLContextSpi** ()
- virtual void **providerInit** (**security::SecureRandom** \*random)=0  
*Perform the initialization of this Context.*
- virtual **SSLParameters** \* **providerGetDefaultSSLParameters** ()  
*Creates a returns a new **SSLParameters** (p. 3551) instance that contains the default settings for this Providers **SSLContext** (p. 3545).*
- virtual **SSLParameters** \* **providerGetSupportedSSLParameters** ()  
*Creates and returns a new **SSLParameters** (p. 3551) instance that contains the full set of supported parameters for this SSL Context.*
- virtual **SocketFactory** \* **providerGetSocketFactory** ()=0  
*Returns a **SocketFactory** (p. 3524) instance that can be used to create new **SSLSocket** (p. 3563) objects.*
- virtual **ServerSocketFactory** \* **providerGetServerSocketFactory** ()=0  
*Returns a **ServerSocketFactory** (p. 3361) instance that can be used to create new **SSLServerSocket** (p. 3554) objects.*

### 6.757.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 3545) provider.

Since:

1.0

### 6.757.2 Constructor & Destructor Documentation

**6.757.2.1** virtual decaf::net::ssl::SSLContextSpi::~~SSLContextSpi () [virtual]

### 6.757.3 Member Function Documentation

**6.757.3.1** virtual **SSLParameters**\* decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters () [virtual]

Creates a returns a new **SSLParameters** (p. 3551) instance that contains the default settings for this Providers **SSLContext** (p. 3545). The returned **SSLParameters** (p. 3551) instance is

requires to have non-empty values in its ciphersuites and protocols.

**Returns:**

new **SSLParameters** (p. 3551) instance with the **SSLContext** (p. 3545) defaults.

**Exceptions:**

*UnsupportedOperationException* if the defaults cannot be obtained.

**6.757.3.2** `virtual ServerSocketFactory* decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory ()` [pure virtual]

Returns a **ServerSocketFactory** (p. 3361) instance that can be used to create new **SSLServerSocket** (p. 3554) objects. The **ServerSocketFactory** (p. 3361) is owned by the Service Provider and should not be destroyed by the caller.

**Returns:**

**SocketFactory** (p. 3524) instance that can be used to create new SSLServerSockets.

**Exceptions:**

*IllegalStateException* if the **SSLContextSpi** (p. 3548) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2843).

**6.757.3.3** `virtual SocketFactory* decaf::net::ssl::SSLContextSpi::providerGetSocketFactory ()` [pure virtual]

Returns a **SocketFactory** (p. 3524) instance that can be used to create new **SSLSocket** (p. 3563) objects. The **SocketFactory** (p. 3524) is owned by the Service Provider and should not be destroyed by the caller.

**Returns:**

**SocketFactory** (p. 3524) instance that can be used to create new SSLSockets.

**Exceptions:**

*IllegalStateException* if the **SSLContextSpi** (p. 3548) object requires initialization but has not been initialized yet.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2843).

**6.757.3.4** `virtual SSLParameters* decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ()` [virtual]

Creates and returns a new **SSLParameters** (p. 3551) instance that contains the full set of supported parameters for this SSL Context. The returned **SSLParameters** (p. 3551) instance requires to have non-empty values in its ciphersuites and protocols.

**Returns:**

a new **SSLParameters** (p. 3551) instance with the full set of settings that are supported.

**Exceptions:**

*UnsupportedOperationException* if the supported parameters cannot be obtained.

**6.757.3.5** **virtual void decaf::net::ssl::SSLContextSpi::providerInit**  
(security::SecureRandom \* *random*) [pure virtual]

Perform the initialization of this Context.

**Parameters:**

*random* Pointer to an instance of a secure random number generator.

**Exceptions:**

*NullPointerException* if the SecureRandom instance is NULL.

*KeyManagementException* if an error occurs while initializing the context.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2844).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLContextSpi.h

## 6.758 decaf::net::ssl::SSLParameters Class Reference

```
#include <src/main/decaf/net/ssl/SSLParameters.h>
```

### Public Member Functions

- **SSLParameters** ()

*Creates a new **SSLParameters** (p. 3551) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.*

- **SSLParameters** (const std::vector< std::string > &cipherSuites)

*Creates a new **SSLParameters** (p. 3551) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.*

- **SSLParameters** (const std::vector< std::string > &cipherSuites, const std::vector< std::string > &protocols)

*Creates a new **SSLParameters** (p. 3551) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.*

- virtual ~**SSLParameters** ()

- std::vector< std::string > **getCipherSuites** () const

- void **setCipherSuites** (const std::vector< std::string > &cipherSuites)

*Sets the vector of ciphersuites.*

- std::vector< std::string > **getProtocols** () const

- void **setProtocols** (const std::vector< std::string > &protocols)

*Sets the vector of protocols.*

- bool **getWantClientAuth** () const

- void **setWantClientAuth** (bool wantClientAuth)

*Sets whether client authentication should be requested.*

- bool **getNeedClientAuth** () const

- void **setNeedClientAuth** (bool needClientAuth)

*Sets whether client authentication should be required.*

### 6.758.1 Constructor & Destructor Documentation

#### 6.758.1.1 decaf::net::ssl::SSLParameters::SSLParameters ()

Creates a new **SSLParameters** (p. 3551) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

#### 6.758.1.2 decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites)

Creates a new **SSLParameters** (p. 3551) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

**Parameters:**

*cipherSuites* The vector of cipherSuites for this **SSLParameters** (p. 3551) instance (can be empty).

**6.758.1.3** `decaf::net::ssl::SSLParameters::SSLParameters (const std::vector< std::string > & cipherSuites, const std::vector< std::string > & protocols)`

Creates a new **SSLParameters** (p. 3551) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

**Parameters:**

*cipherSuites* The vector of cipherSuites for this **SSLParameters** (p. 3551) instance (can be empty).

*protocols* The vector of protocols for this **SSLParameters** (p. 3551) instance (can be empty).

**6.758.1.4** `virtual decaf::net::ssl::SSLParameters::~~SSLParameters () [virtual]`

**6.758.2 Member Function Documentation**

**6.758.2.1** `std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites () const [inline]`

**Returns:**

a copy of the vector of ciphersuites or an empty vector if none have been set.

**6.758.2.2** `bool decaf::net::ssl::SSLParameters::getNeedClientAuth () const [inline]`

**Returns:**

whether client authentication should be required.

**6.758.2.3** `std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols () const [inline]`

**Returns:**

a copy of the vector of protocols or an empty vector if none have been set.

**6.758.2.4** `bool decaf::net::ssl::SSLParameters::getWantClientAuth () const [inline]`

**Returns:**

whether client authentication should be requested.



**6.758.2.5** void decaf::net::ssl::SSLParameters::setCipherSuites (const std::vector< std::string > & *cipherSuites*) [inline]

Sets the vector of ciphersuites.

**Parameters:**

*cipherSuites* The vector of cipherSuites (can be an empty vector).

**6.758.2.6** void decaf::net::ssl::SSLParameters::setNeedClientAuth (bool *needClientAuth*) [inline]

Sets whether client authentication should be required. Calling this method clears the wantClientAuth flag.

**Parameters:**

*needClientAuth* whether client authentication should be required.

**6.758.2.7** void decaf::net::ssl::SSLParameters::setProtocols (const std::vector< std::string > & *protocols*) [inline]

Sets the vector of protocols.

**Parameters:**

*protocols* the vector of protocols (or an empty vector)

**6.758.2.8** void decaf::net::ssl::SSLParameters::setWantClientAuth (bool *wantClientAuth*) [inline]

Sets whether client authentication should be requested. Calling this method clears the needClientAuth flag.

**Parameters:**

*whether* client authentication should be requested.

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLParameters.h

## 6.759 decaf::net::ssl::SSLServerSocket Class Reference

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

#include <src/main/decaf/net/ssl/SSLServerSocket.h> Inheritance diagram for decaf::net::ssl::SSLServerSocket:

### Public Member Functions

- virtual `~SSLServerSocket ()`
- virtual `std::vector< std::string > getSupportedCipherSuites () const =0`  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3554).*
- virtual `std::vector< std::string > getSupportedProtocols () const =0`  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3554) instance.*
- virtual `std::vector< std::string > getEnabledCipherSuites () const =0`  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3554).*
- virtual void `setEnabledCipherSuites (const std::vector< std::string > &suites)=0`  
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3554) connection.*
- virtual `std::vector< std::string > getEnabledProtocols () const =0`  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3554).*
- virtual void `setEnabledProtocols (const std::vector< std::string > &protocols)=0`  
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3554) connection.*
- virtual bool `getWantClientAuth () const =0`
- virtual void `setWantClientAuth (bool value)=0`  
*Sets whether or not this **Socket** (p. 3503) will request Client Authentication.*
- virtual bool `getNeedClientAuth () const =0`
- virtual void `setNeedClientAuth (bool value)=0`  
*Sets whether or not this **Socket** (p. 3503) will require Client Authentication.*

### Protected Member Functions

- `SSLServerSocket ()`  
*Creates a non-bound server socket.*
- `SSLServerSocket (int port)`

*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*

- **SSLServerSocket** (int port, int backlog)

*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*

- **SSLServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

*Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen.*

### 6.759.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol. The main function of this class is to create **SSLSocket** (p. 3563) objects by accepting connections from client sockets over SSL.

Since:

1.0

### 6.759.2 Constructor & Destructor Documentation

#### 6.759.2.1 decaf::net::ssl::SSLServerSocket::SSLServerSocket () [protected]

Creates a non-bound server socket.

#### 6.759.2.2 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int port) [protected]

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3537) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

Exceptions:

***IOException*** if there is an I/O error while performing this operation.

***IllegalArgumentException*** if the port value is negative or greater than 65535.

#### 6.759.2.3 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int port, int backlog) [protected]

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. When this constructor is called the size of the backlog queue is set at

backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3537) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

#### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The the number of incoming connection attempts to queue before connections are refused.

#### Exceptions:

*IOException* if there is an I/O error while performing this operation.

*IllegalArgumentException* if the port value is negative or greater than 65535.

#### 6.759.2.4 decaf::net::ssl::SSLServerSocket::SSLServerSocket (int *port*, int *backlog*, const decaf::net::InetAddress \* *address*) [protected]

Creates a new **ServerSocket** (p. 3352) bound to the specified port, if the value of port is 0, then any free port is chosen. If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3537) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3529) is created.

#### Parameters:

*port* The port to bind the **ServerSocket** (p. 3352) to.

*backlog* The the number of incoming connection attempts to queue before connections are refused.

*ifAddress* The IP Address to bind to on the local machine.

#### Exceptions:

*IOException* if there is an I/O error while performing this operation.

*IllegalArgumentException* if the port value is negative or greater than 65535.

#### 6.759.2.5 virtual decaf::net::ssl::SSLServerSocket::~~SSLServerSocket () [virtual]

### 6.759.3 Member Function Documentation

#### 6.759.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites () const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3554).

**Returns:**

vector of the names of all enabled Cipher Suites.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2849).

**6.759.3.2** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledProtocols () const [pure virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3554).

**Returns:**

vector of the names of all enabled Protocols.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2850).

**6.759.3.3** `virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth () const [pure virtual]`

**Returns:**

true if the **Socket** (p. 3503) requires client Authentication.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2850).

**6.759.3.4** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedCipherSuites () const [pure virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3554). Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).

**Returns:**

a vector containing the names of all the supported cipher suites.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2850).

**6.759.3.5** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getSupportedProtocols () const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3554) instance.

**Returns:**

a vector containing the names of all the supported protocols.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2850).

**6.759.3.6** `virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth () const`  
[pure virtual]

**Returns:**

true if the **Socket** (p. 3503) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2851).

**6.759.3.7** `virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites`  
(const std::vector< std::string > & *suites*) [pure virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3554) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters:**

*suites* An Vector of names for all the Cipher Suites that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2851).

**6.759.3.8** `virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols` (const  
std::vector< std::string > & *protocols*) [pure virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3554) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters:**

*protocols* An Vector of names for all the Protocols that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2851).

**6.759.3.9** `virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth` (bool  
*value*) [pure virtual]

Sets whether or not this **Socket** (p. 3503) will require Client Authentication. If set to true the **Socket** (p. 3503) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

**Parameters:**

*value* Whether the server socket should require client authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2851).

**6.759.3.10 virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth (bool *value*) [pure virtual]**

Sets whether or not this **Socket** (p. 3503) will request Client Authentication. If set to true the **Socket** (p. 3503) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

**Parameters:**

*value* Whether the server socket should request client authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2852).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocket.h**

## 6.760 decaf::net::ssl::SSLServerSocketFactory Class Reference

Factory class interface that provides methods to create SSL Server Sockets.

`#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>`Inheritance diagram for decaf::net::ssl::SSLServerSocketFactory:

### Public Member Functions

- virtual `~SSLServerSocketFactory ()`
- virtual `std::vector< std::string > getDefaultCipherSuites ()=0`

*Returns the list of cipher suites which are enabled by default.*

- virtual `std::vector< std::string > getSupportedCipherSuites ()=0`

*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*

### Static Public Member Functions

- static `ServerSocketFactory * getDefault ()`

*Returns the current default SSL **ServerSocketFactory** (p. 3361), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

### Protected Member Functions

- `SSLServerSocketFactory ()`

#### 6.760.1 Detailed Description

Factory class interface that provides methods to create SSL Server Sockets.

**Since:**

1.0



## 6.760.2 Constructor & Destructor Documentation

**6.760.2.1** `decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory ()`  
[protected]

**6.760.2.2** `virtual decaf::net::ssl::SSLServerSocketFactory::~~SSLServerSocketFactory ()` [virtual]

## 6.760.3 Member Function Documentation

**6.760.3.1** `static ServerSocketFactory* decaf::net::ssl::SSLServerSocketFactory::getDefault ()`  
[static]

Returns the current default SSL **ServerSocketFactory** (p.3361), the factory is returned as a pointer however the caller does not own this pointer and should not delete it. This method returns **SSLContext::getDefault()** (p.3545)->**getServerSocketFactory()**. If that call fails, a non-functional factory is returned.

### Returns:

the default SSL **ServerSocketFactory** (p.3361) pointer.

### See also:

**decaf::net::ssl::SSLContext::getDefault()** (p.3545)

Reimplemented from **decaf::net::ServerSocketFactory** (p.3363).

**6.760.3.2** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites ()` [pure virtual]

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

### Returns:

an STL vector containing the list of cipher suites enabled by default.

### See also:

**getSupportedCipherSuites()** (p.3561)

Implemented in **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p.1695), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p.2856).

**6.760.3.3** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getSupportedCipherSuites ()`  
[pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include

cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns:**

an STL vector containing the list of supported cipher suites.

**See also:**

**getDefaultCipherSuites()** (p. 3561)

Implemented in **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1696), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2857).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocketFactory.h`

## 6.761 decaf::net::ssl::SSLSocket Class Reference

#include <src/main/decaf/net/ssl/SSLSocket.h> Inheritance diagram for decaf::net::ssl::SSLSocket:

### Public Member Functions

- **SSLSocket** ()
- **SSLSocket** (const **InetAddress** \*address, int port)  
*Creates a new **SSLSocket** (p. 3563) instance and connects it to the given address and port.*
- **SSLSocket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **SSLSocket** (p. 3563) instance and connects it to the given address and port.*
- **SSLSocket** (const std::string &host, int port)  
*Creates a new **SSLSocket** (p. 3563) instance and connects it to the given host and port.*
- **SSLSocket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **SSLSocket** (p. 3563) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3563).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3563) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3503).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0  
*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3503) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3503).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0  
*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3503) connection.*
- virtual **SSLParameters** **getSSLParameters** () const  
*Returns an **SSLParameters** (p. 3551) object for this **SSLSocket** (p. 3563) instance.*

- virtual void **setSSLParameters** (const **SSLParameters** &value)  
*Sets the **SSLParameters** (p. 3551) for this **SSLSocket** (p. 3563) using the supplied **SSLParameters** (p. 3551) instance.*
- virtual void **startHandshake** ()=0  
*Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*
- virtual void **setUseClientMode** (bool value)=0  
*Determines the mode that the socket uses when a handshake is initiated, client or server.*
- virtual bool **getUseClientMode** () const =0  
*Gets whether this **Socket** (p. 3503) is in Client or Server mode, true indicates that the mode is set to Client.*
- virtual void **setNeedClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3503) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getNeedClientAuth** () const =0  
*Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.*
- virtual void **setWantClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3503) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getWantClientAuth** () const =0  
*Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.*

### 6.761.1 Detailed Description

Since:

1.0

### 6.761.2 Constructor & Destructor Documentation

#### 6.761.2.1 `decaf::net::ssl::SSLSocket::SSLSocket ()`

#### 6.761.2.2 `decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress * address, int port)`

Creates a new **SSLSocket** (p. 3563) instance and connects it to the given address and port. If the host parameter is empty then the loop back address is used.

Parameters:

***address*** The address to connect to.

*port* The port number to connect to [0...65535]

**Exceptions:**

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*NullPointerException* if the **InetAddress** (p. 2016) instance is NULL.

*IllegalArgumentException* if the port is not in range [0...65535]

**6.761.2.3 decaf::net::ssl::SSLSocket::SSLSocket (const InetAddress \* address, int port, const InetAddress \* localAddress, int localPort)**

Creates a new **SSLSocket** (p. 3563) instance and connects it to the given address and port. The **Socket** (p. 3503) will also bind to the local address and port specified.

**Parameters:**

*address* The address to connect to.

*port* The port number to connect to [0...65535]

*localAddress* The IP address on the local machine to bind to.

*localPort* The port on the local machine to bind to.

**Exceptions:**

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*NullPointerException* if the **InetAddress** (p. 2016) instance is NULL.

*IllegalArgumentException* if the port is not in range [0...65535]

**6.761.2.4 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & host, int port)**

Creates a new **SSLSocket** (p. 3563) instance and connects it to the given host and port. If the host parameter is empty then the loop back address is used.

**Parameters:**

*host* The host name or IP address to connect to, empty string means loopback.

*port* The port number to connect to [0...65535]

**Exceptions:**

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*IllegalArgumentException* if the port is not in range [0...65535]

### 6.761.2.5 decaf::net::ssl::SSLSocket::SSLSocket (const std::string & *host*, int *port*, const InetAddress \* *localAddress*, int *localPort*)

Creates a new **SSLSocket** (p. 3563) instance and connects it to the given host and port. If the host parameter is empty then the loop back address is used.

#### Parameters:

*host* The host name or IP address to connect to, empty string means loopback.

*port* The port number to connect to [0...65535]

*localAddress* The IP address on the local machine to bind to.

*localPort* The port on the local machine to bind to.

#### Exceptions:

*UnknownHostException* (p. 3907) if the host cannot be resolved.

*IOException* if an I/O error occurs while connecting the **Socket** (p. 3503).

*IllegalArgumentException* if the port is not in range [0...65535]

### 6.761.2.6 virtual decaf::net::ssl::SSLSocket::~SSLSocket () [virtual]

## 6.761.3 Member Function Documentation

### 6.761.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledCipherSuites () const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3503).

#### Returns:

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2863).

### 6.761.3.2 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledProtocols () const [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3503).

#### Returns:

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2864).

**6.761.3.3 virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth () const**  
[pure virtual]

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected. This option is only useful when the socket is operating in server mode.

**Returns:**

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2864).

**6.761.3.4 virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters () const** [virtual]

Returns an **SSLParameters** (p. 3551) object for this **SSLSocket** (p. 3563) instance. The cipher-Suites and protocols vectors in the returned **SSLParameters** (p. 3551) reference will never be empty.

**Returns:**

an **SSLParameters** (p. 3551) object with the settings in use for the **SSLSocket** (p. 3563).

**6.761.3.5 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedCipherSuites () const** [pure virtual]

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3563). Normally not all of these cipher suites will be enabled on the **Socket** (p. 3503).

**Returns:**

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2865).

**6.761.3.6 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedProtocols () const** [pure virtual]

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3563) instance.

**Returns:**

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2865).

**6.761.3.7** virtual bool decaf::net::ssl::SSLSocket::getUseClientMode () const [pure virtual]

Gets whether this **Socket** (p. 3503) is in Client or Server mode, true indicates that the mode is set to Client.

**Returns:**

true if the **Socket** (p. 3503) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2865).

**6.761.3.8** virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth () const [pure virtual]

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication. This option is only useful when the socket is operating in server mode.

**Returns:**

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2866).

**6.761.3.9** virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites (const std::vector< std::string > & suites) [pure virtual]

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3503) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters:**

*suites* An Vector of names for all the Cipher Suites that are to be enabled.

**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2867).

**6.761.3.10** virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols (const std::vector< std::string > & protocols) [pure virtual]

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3503) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters:**

*protocols* An Vector of names for all the Protocols that are to be enabled.



**Exceptions:**

*IllegalArgumentException* if the vector is empty or one of the names is invalid.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2867).

**6.761.3.11 virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth (bool *value*)**  
[pure virtual]

Sets the **Socket** (p. 3503) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket. This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the `setWantClientAuth` method.

**Parameters:**

*value* The value indicating if a client is required to authenticate itself or not.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2867).

**6.761.3.12 virtual void decaf::net::ssl::SSLSocket::setSSLParameters (const SSLParameters & *value*)** [virtual]

Sets the **SSLParameters** (p. 3551) for this **SSLSocket** (p. 3563) using the supplied **SSLParameters** (p. 3551) instance. If the cipherSuites vector in the **SSLParameters** (p. 3551) instance is not empty then the `setEnabledCipherSuites` method is called with that vector, if the protocols vector in the **SSLParameters** (p. 3551) instance is not empty then the `setEnabledProtocols` method is called with that vector. If the `needClientAuth` value or the `wantClientAuth` value is true then the `setNeedClientAuth` and `setWantClientAuth` methods are called respectively with a value of true, otherwise the `setWantClientAuth` method is called with a value of false.

**Parameters:**

*value* The **SSLParameters** (p. 3551) instance that is used to update this **SSLSocket**'s settings.

**Exceptions:**

*IllegalArgumentException* if an error occurs while calling `setEnabledCipherSuites` or `setEnabledProtocols`.

**6.761.3.13 virtual void decaf::net::ssl::SSLSocket::setUseClientMode (bool *value*)**  
[pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server. This method must be called prior to any handshake attempts on this **Socket** (p. 3503), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

**Parameters:**

*value* The mode setting, true for client or false for server.

**Exceptions:**

*IllegalArgumentException* if the handshake process has begun and mode is locked.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2868).

**6.761.3.14** `virtual void decaf::net::ssl::SSLSocket::setWantClientAuth (bool value)`  
[pure virtual]

Sets the **Socket** (p. 3503) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket. This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

**Parameters:**

*value* The value indicating if a client is requested to authenticate itself or not.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2868).

**6.761.3.15** `virtual void decaf::net::ssl::SSLSocket::startHandshake ()` [pure virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session. When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

**Exceptions:**

*IOException* if an I/O error occurs while performing the Handshake

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2869).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocket.h`

## 6.762 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a **SocketFactory** (p. 3524) that can create **SSLSocket** (p. 3563) objects.

#include <src/main/decaf/net/ssl/SSLSocketFactory.h>Inheritance diagram for decaf::net::ssl::SSLSocketFactory:

### Public Member Functions

- virtual **~SSLSocketFactory** ()
- virtual std::vector< std::string > **getDefaultCipherSuites** ()=0  
*Returns the list of cipher suites which are enabled by default.*
- virtual std::vector< std::string > **getSupportedCipherSuites** ()=0  
*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*
- virtual **Socket** \* **createSocket** (**Socket** \*socket, std::string host, int port, bool autoClose)=0  
*Returns a socket layered over an existing socket connected to the named host, at the given port.*

### Static Public Member Functions

- static **SocketFactory** \* **getDefault** ()  
*Returns the current default SSL **SocketFactory** (p. 3524), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

### Protected Member Functions

- **SSLSocketFactory** ()

#### 6.762.1 Detailed Description

Factory class interface for a **SocketFactory** (p. 3524) that can create **SSLSocket** (p. 3563) objects.

**Since:**

1.0

## 6.762.2 Constructor & Destructor Documentation

**6.762.2.1** `decaf::net::ssl::SSLSocketFactory::SSLSocketFactory ()` [protected]

**6.762.2.2** `virtual decaf::net::ssl::SSLSocketFactory::~~SSLSocketFactory ()`  
[virtual]

## 6.762.3 Member Function Documentation

**6.762.3.1** `virtual Socket* decaf::net::ssl::SSLSocketFactory::createSocket (Socket *  
socket, std::string host, int port, bool autoClose)` [pure virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port. This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

### Parameters:

*socket* The existing socket to layer over.

*host* The server host the original **Socket** (p. 3503) is connected to.

*port* The server port the original **Socket** (p. 3503) is connected to.

*autoClose* Should the layered over **Socket** (p. 3503) be closed when the topmost socket is closed.

### Returns:

a new **Socket** (p. 3503) instance that wraps the given **Socket** (p. 3503).

### Exceptions:

*IOException* if an I/O exception occurs while performing this operation.

*UnknownHostException* (p. 3907) if the host is unknown.

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p. 1699), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2877).

**6.762.3.2** `static SocketFactory* decaf::net::ssl::SSLSocketFactory::getDefault ()`  
[static]

Returns the current default SSL **SocketFactory** (p. 3524), the factory is returned as a pointer however the caller does not own this pointer and should not delete it. This method returns `SSLContext::getDefault()` (p. 3545)->getSocketFactory(). If that call fails, a non-functional factory is returned.

### Returns:

the default SSL **SocketFactory** (p. 3524) pointer.

### See also:

`decaf::net::ssl::SSLContext::getDefault()` (p. 3545)

Reimplemented from `decaf::net::SocketFactory` (p. 3527).

### 6.762.3.3 `virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getDefaultCipherSuites ()` [pure virtual]

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns:

an STL vector containing the list of cipher suites enabled by default.

#### See also:

`getSupportedCipherSuites()` (p. 3573)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p.1702), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2880).

### 6.762.3.4 `virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getSupportedCipherSuites ()` [pure virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

#### Returns:

an STL vector containing the list of supported cipher suites.

#### See also:

`getDefaultCipherSuites()` (p. 3573)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p.1703), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2880).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocketFactory.h`

## 6.763 activemq::transport::tcp::SslTransport Class Reference

**Transport** (p. 3883) for connecting to a Broker using an SSL Socket.

#include <src/main/activemq/transport/tcp/SslTransport.h> Inheritance diagram for activemq::transport::tcp::SslTransport:

### Public Member Functions

- **SslTransport** (const **Pointer**< **Transport** > &next)  
*Creates a new instance of the **SslTransport** (p. 3574), the **transport** (p. 97) will not attempt to connect to a remote host until the connect method is called.*
- virtual ~**SslTransport** ()

### Protected Member Functions

- virtual **decaf::net::Socket** \* **createSocket** ()  
*Create an unconnected Socket instance to be used by the **transport** (p. 97) to communicate with the broker.*  
**Returns:**  
*a newly created unconnected Socket instance.*  
**Exceptions:**  
***IOException** if there is an error while creating the unconnected Socket.*
- virtual void **configureSocket** (**decaf::net::Socket** \*socket, **decaf::util::Properties** &properties)

### 6.763.1 Detailed Description

**Transport** (p. 3883) for connecting to a Broker using an SSL Socket. This **transport** (p. 97) simply wraps the **TcpTransport** (p. 3752) and provides the **TcpTransport** (p. 3752) an SSL based Socket pointer allowing the **core** (p. 89) **TcpTransport** (p. 3752) logic to be reused.

Since:

3.2.0

### 6.763.2 Constructor & Destructor Documentation

#### 6.763.2.1 activemq::transport::tcp::SslTransport::SslTransport (const **Pointer**< **Transport** > & next)

Creates a new instance of the **SslTransport** (p. 3574), the **transport** (p. 97) will not attempt to connect to a remote host until the connect method is called.

**Parameters:**

*next* the next **transport** (p. 97) in the chain

**6.763.2.2** virtual **activemq::transport::tcp::SslTransport::~~SslTransport** ()  
[virtual]

**6.763.3 Member Function Documentation**

**6.763.3.1** virtual void **activemq::transport::tcp::SslTransport::configureSocket**  
(**decaf::net::Socket** \* *socket*, **decaf::util::Properties** & *properties*)  
[protected, virtual]

**6.763.3.2** virtual **decaf::net::Socket\*** **activemq::transport::tcp::SslTransport::createSocket** ()  
[protected, virtual]

Create an unconnected **Socket** instance to be used by the **transport** (p. 97) to communicate with the broker.

**Returns:**

a newly created unconnected **Socket** instance.

**Exceptions:**

***IOException*** if there is an error while creating the unconnected **Socket**.

Reimplemented from **activemq::transport::tcp::TcpTransport** (p. 3754).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransport.h`

## 6.764 activemq::transport::tcp::SslTransportFactory Class Reference

#include <src/main/activemq/transport/tcp/SslTransportFactory.h> Inheritance diagram for activemq::transport::tcp::SslTransportFactory:

### Public Member Functions

- virtual `~SslTransportFactory ()`

### Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw ( exceptions::ActiveMQException )`  
*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

### 6.764.1 Constructor & Destructor Documentation

- 6.764.1.1 virtual  
`activemq::transport::tcp::SslTransportFactory::~~SslTransportFactory ()`  
 [virtual]

### 6.764.2 Member Function Documentation

- 6.764.2.1 virtual `Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite (const decaf::net::URI & location, const Pointer< wireformat::WireFormat > & wireFormat, const decaf::util::Properties & properties) throw ( exceptions::ActiveMQException )` [protected, virtual]

Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.

#### Parameters:

*location* - URI location to connect to.

*wireFormat* - the assigned WireFormat for the new **Transport** (p. 3883).

*properties* - Properties to apply to the **transport** (p. 97).

#### Returns:

new Pointer to a **SslTransport** (p. 3574).

#### Exceptions:

*ActiveMQException* if an error occurs



Reimplemented from **activemq::transport::tcp::TcpTransportFactory** (p. 3757).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransportFactory.h`

## 6.765 activemq::commands::BrokerError::StackTraceElement Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

### Data Fields

- `std::string` **ClassName**
- `std::string` **FileName**
- `std::string` **MethodName**
- `int` **LineNumber**

### 6.765.1 Field Documentation

**6.765.1.1** `std::string` `activemq::commands::BrokerError::StackTraceElement::ClassName`

**6.765.1.2** `std::string` `activemq::commands::BrokerError::StackTraceElement::FileName`

**6.765.1.3** `int` `activemq::commands::BrokerError::StackTraceElement::LineNumber`

**6.765.1.4** `std::string` `activemq::commands::BrokerError::StackTraceElement::MethodName`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

## 6.766 decaf::internal::io::StandardErrorOutputStream Class Reference

Wrapper Around the Standard error Output facility on the current platform.

`#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>`Inheritance diagram for decaf::internal::io::StandardErrorOutputStream:

### Public Member Functions

- **StandardErrorOutputStream** ()
- virtual **~StandardErrorOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )

*Flushes this stream by writing any buffered output to the underlying stream.*

**Exceptions:**

*IOException (p. 2142) if an I/O error occurs.*

*The default implementation of this method does nothing.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

**Exceptions:**

*IOException (p. 2142) if an error occurs while closing.*

*The default implementation of this method does nothing.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

#### 6.766.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

## 6.766.2 Constructor & Destructor Documentation

**6.766.2.1** `decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream()`

**6.766.2.2** `virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream()` [virtual]

## 6.766.3 Member Function Documentation

**6.766.3.1** `virtual void decaf::internal::io::StandardErrorOutputStream::close()`  
`throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from `decaf::io::OutputStream` (p. 2909).

**6.766.3.2** `virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded`  
`(const unsigned char * buffer, int size, int offset, int length) throw (`  
`decaf::io::IOException, decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2909).

**6.766.3.3** `virtual void decaf::internal::io::StandardErrorOutputStream::doWriteByte`  
`(unsigned char value) throw ( decaf::io::IOException )` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2910).

**6.766.3.4** `virtual void decaf::internal::io::StandardErrorOutputStream::flush()`  
`throw ( decaf::io::IOException )` [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2910).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardErrorOutputStream.h`

## 6.767 decaf::internal::io::StandardInputStream Class Reference

#include <src/main/decaf/internal/io/StandardInputStream.h> Inheritance diagram for decaf::internal::io::StandardInputStream:

### Public Member Functions

- **StandardInputStream** ()
- virtual **~StandardInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

### Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )

### 6.767.1 Constructor & Destructor Documentation

**6.767.1.1** decaf::internal::io::StandardInputStream::StandardInputStream ()

**6.767.1.2** virtual decaf::internal::io::StandardInputStream::~~StandardInputStream () [virtual]

### 6.767.2 Member Function Documentation

**6.767.2.1** virtual int decaf::internal::io::StandardInputStream::available () const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available. The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns:

the number of bytes available on this input stream.

#### Exceptions:

**IOException** if an I/O error occurs.

Reimplemented from **decaf::io::InputStream** (p. 2045).

**6.767.2.2** virtual int decaf::internal::io::StandardInputStream::doReadByte ()  
throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 2046).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardInputStream.h**

## 6.768 decaf::internal::io::StandardOutputStream Class Reference

#include <src/main/decaf/internal/io/StandardOutputStream.h> Inheritance diagram for decaf::internal::io::StandardOutputStream:

### Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*  
**Exceptions:**  
*IOException* (p. 2142) if an I/O error occurs.  
The default implementation of this method does nothing.
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*  
*The object is generally no longer usable after calling close.*  
**Exceptions:**  
*IOException* (p. 2142) if an error occurs while closing.  
The default implementation of this method does nothing.

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.768.1 Constructor & Destructor Documentation

**6.768.1.1** decaf::internal::io::StandardOutputStream::StandardOutputStream ()

**6.768.1.2** virtual  
decaf::internal::io::StandardOutputStream::~~StandardOutputStream ()  
[virtual]

## 6.768.2 Member Function Documentation

**6.768.2.1** virtual void decaf::internal::io::StandardOutputStream::close () throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.



**Exceptions:**

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2909).

**6.768.2.2** virtual void decaf::internal::io::StandardOutputStream::doWriteArrayBounded (const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2909).

**6.768.2.3** virtual void decaf::internal::io::StandardOutputStream::doWriteByte (unsigned char *value*) throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2910).

**6.768.2.4** virtual void decaf::internal::io::StandardOutputStream::flush () throw ( decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2910).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

## 6.769 cms::Startable Class Reference

Interface for a class that implements the start method.

#include <src/main/cms/Startable.h> Inheritance diagram for cms::Startable:

### Public Member Functions

- virtual `~Startable()`
- virtual void `start()` throw ( CMSEException )  
*Starts the service.*

### 6.769.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p. 3586) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since:

1.0

### 6.769.2 Constructor & Destructor Documentation

**6.769.2.1** virtual cms::Startable::~~Startable() [inline, virtual]

### 6.769.3 Member Function Documentation

**6.769.3.1** virtual void cms::Startable::start() throw ( CMSEException ) [pure virtual]

Starts the service.

Exceptions:

*CMSEException* (p. 1160) if an internal error occurs while starting.

Implemented in **activemq::core::ActiveMQConnection** (p. 290).

The documentation for this class was generated from the following file:

- src/main/cms/Startable.h

## 6.770 decaf::lang::STATIC\_CAST\_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.771 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

### Public Member Functions

- **StaticInitializer** ()
- virtual **~StaticInitializer** ()

### Static Public Attributes

- static std::string **destOptions** [NUM\_OPTIONS]
- static std::string **uriParams** [NUM\_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

### 6.771.1 Constructor & Destructor Documentation

**6.771.1.1** **activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer** ()

**6.771.1.2** virtual  
**activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer**  
() [inline, virtual]

### 6.771.2 Field Documentation

**6.771.2.1** std::map<std::string, **DestinationOption**> **activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap**  
[static]

**6.771.2.2** std::string  
**activemq::core::ActiveMQConstants::StaticInitializer::destOptions**[NUM\_OPTIONS] [static]

**6.771.2.3** std::string  
**activemq::core::ActiveMQConstants::StaticInitializer::uriParams**[NUM\_PARAMS] [static]

**6.771.2.4** std::map<std::string, **URIParam**> **activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap**  
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

## 6.772 decaf::util::StlList< E > Class Template Reference

**List** (p.2337) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

#include <src/main/decaf/util/StlList.h>Inheritance diagram for decaf::util::StlList< E >:

### Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

### Public Member Functions

- **StlList** ()  
*Default constructor - does nothing.*
- **StlList** (const **StlList** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlList** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual ~**StlList** ()
- virtual bool **equals** (const **StlList** &source) const  
*Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.*  
**Returns:**  
*true if the Collections contain the same elements.*
- virtual **Iterator**< E > \* **iterator** ()  
**Returns:**  
*an iterator over a set of elements of type T.*
- virtual **Iterator**< E > \* **iterator** () const
- virtual **ListIterator**< E > \* **listIterator** ()  
**Returns:**  
*a list iterator over the elements in this list (in proper sequence).*
- virtual **ListIterator**< E > \* **listIterator** () const
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
**Parameters:**  
*index index of first element to be returned from the list iterator (by a call to the next method).*

**Returns:**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to `next`. An initial call to `previous` would return the element with the specified index minus one.

**Exceptions:**

**IndexOutOfBoundsException** if the index is out of range (`index < 0 || index > size()` (p. 1192))

- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const throw ( default::lang::exceptions::IndexOutOfBoundsException )
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all of the elements from this collection (optional operation).*

*The collection will be empty after this method returns.*

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2155) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

**Exceptions:**

**UnsupportedOperationException** if the clear operation is not supported by this collection

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

*This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

**Parameters:**

**value** - the value whose presence is to be queried for in this **Collection** (p. 1184).

**Returns:**

*true if the value is contained in this collection*

**Exceptions:**

**Exception** if an error occurs,

- virtual std::size\_t **indexOf** (const E &value) throw ( default::lang::exceptions::NoSuchElementException )

*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*

*More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.*

**Parameters:**

**value** - element to search for

**Returns:**

*the index of the first occurrence of the specified element in this list,*

**Exceptions:**

**NoSuchElementException** if value is not in the list

- virtual std::size\_t **lastIndexOf** (const E &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.*

*More formally, returns the highest index  $i$  such that  $get(i) == value$  or -1 if there is no such index.*

**Parameters:**

*value* - element to search for

**Returns:**

*the index of the last occurrence of the specified element in this list.*

**Exceptions:**

*NoSuchElementException* if value is not in the list

- virtual bool **isEmpty** () const

*Returns true if this collection contains no elements.*

*This implementation returns **size()** (p. 1192) == 0.*

**Returns:**

*true if the size method return 0.*

- virtual std::size\_t **size** () const

*Returns the number of elements in this collection.*

*If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.*

**Returns:**

*the number of elements in this collection*

- virtual E **get** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Gets the element contained at position passed.*

**Parameters:**

*index* - position to get

**Returns:**

*value at index*

- virtual E **set** (std::size\_t index, const E &element) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Replaces the element at the specified position in this list with the specified element.*

**Parameters:**

*index* - index of the element to replace

*element* - element to be stored at the specified position

**Returns:**

*the element previously at the specified position*

**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size

- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1184) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

**Parameters:**

*value* - reference to the element to add.

**Returns:**

true if the element was added

**Exceptions:**

**UnsupportedOperationException**

**IllegalArgumentException**

**IllegalStateException** if the element cannot be added at this time due to insertion restrictions

- virtual void **add** (std::size\_t index, const E &element) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException )

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Parameters:**

*index* - index at which the specified element is to be inserted

*element* - element to be inserted

**Exceptions:**

**IndexOutOfBoundsException** - if the index is greater than size

**UnsupportedOperationException** - If the collection is non-modifiable.

- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

**Parameters:**

*index* The index at which to insert the first element from the specified collection

*source* The **Collection** (p. 1184) containing elements to be added to this list

**Returns:**

true if this list changed as a result of the call

**Exceptions:**

**IndexOutOfBoundsException** - if the index is greater than size

**UnsupportedOperationException** - If the collection is non-modifiable.



- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

**Parameters:**

*value* - element to be removed from this collection, if present

**Returns:**

*true if an element was removed as a result of this call*

**Exceptions:**

**UnsupportedOperationException** if the remove operation is not supported by this collection.

**IllegalArgumentException** If the value is not a valid entry for this **Collection** (p. 1184).

- virtual E **remove** (std::size\_t index) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Removes the element at the specified position in this list.*

*Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.*

**Parameters:**

*index* - the index of the element to be removed

**Returns:**

*the element previously at the specified position*

**Exceptions:**

**IndexOutOfBoundsException** - if the index is greater than size

**UnsupportedOperationException** - If the collection is non-modifiable.

## 6.772.1 Detailed Description

```
template<typename E> class decaf::util::StlList< E >
```

**List** (p. 2337) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

## 6.772.2 Constructor & Destructor Documentation

**6.772.2.1** `template<typename E> decaf::util::StlList< E >::StlList () [inline]`

Default constructor - does nothing.

**6.772.2.2** `template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

**Parameters:**

*source* The source set.

**6.772.2.3** `template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

**Parameters:**

*source* The source set.

**6.772.2.4** `template<typename E> virtual decaf::util::StlList< E >::~StlList () [inline, virtual]`

## 6.772.3 Member Function Documentation

**6.772.3.1** `template<typename E> virtual void decaf::util::StlList< E >::add (std::size_t index, const E & element) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Parameters:**

*index* - index at which the specified element is to be inserted

*element* - element to be inserted

**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size

*UnsupportedOperationException* - If the collection is non-modifiable.

Implements `decaf::util::List< E >` (p. 2338).

**6.772.3.2** `template<typename E> virtual bool decaf::util::StlList< E >::add (const E & value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1184) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters:

*value* - reference to the element to add.

#### Returns:

true if the element was added

#### Exceptions:

*UnsupportedOperationException*

*IllegalArgumentException*

*IllegalStateException* if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1186).

Referenced by decaf::util::StlList< cms::Connection \* >::addAll().

```
6.772.3.3  template<typename E> virtual bool decaf::util::StlList< E
>::addAll (std::size_t index, const Collection< E > & source)
throw ( decaf::lang::exceptions::UnsupportedOperationException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

#### Parameters:

*index* The index at which to insert the first element from the specified collection

*source* The **Collection** (p.1184) containing elements to be added to this list

#### Returns:

true if this list changed as a result of the call

**Exceptions:**

***IndexOutOfBoundsException*** - if the index is greater than size

***UnsupportedOperationException*** - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2339).

**6.772.3.4** `template<typename E> virtual void decaf::util::StlList< E >::clear ()  
throw ( lang::exceptions::UnsupportedOperationException ) [inline,  
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2155) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

**Exceptions:**

***UnsupportedOperationException*** if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 183).

**6.772.3.5** `template<typename E> virtual bool decaf::util::StlList< E >::contains  
(const E & value) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

**Parameters:**

*value* - the value whose presence is to be queried for in this **Collection** (p. 1184).

**Returns:**

true if the value is contained in this collection

**Exceptions:**

***Exception*** if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 184).

**6.772.3.6** `template<typename E> virtual void decaf::util::StlList< E >::copy (const  
StlList< E > & source) [inline, virtual]`

Referenced by **decaf::util::StlList< cms::Connection \* >::StlList()**.

**6.772.3.7** `template<typename E> virtual bool decaf::util::StlList< E >::equals  
(const StlList< E > & source) const` [inline, virtual]

Compares the passed collection to this one, if they contain the same elements, i.e.  
all their elements are equivalent, then it returns true.

**Returns:**

true if the Collections contain the same elements.

Implements **decaf::util::Collection< E >** (p. 1189).

**6.772.3.8** `template<typename E> virtual E decaf::util::StlList< E >::get (std::size_t  
index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
)` [inline, virtual]

Gets the element contained at position passed.

**Parameters:**

*index* - position to get

**Returns:**

value at index

Implements **decaf::util::List< E >** (p. 2339).

**6.772.3.9** `template<typename E> virtual std::size_t decaf::util::StlList<  
E >::indexOf (const E & value) throw ( de-  
caf::lang::exceptions::NoSuchElementException )` [inline,  
virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does  
not contain the element.

More formally, returns the lowest index i such that get(i) == value, or -1 if there is no such index.

**Parameters:**

*value* - element to search for

**Returns:**

the index of the first occurrence of the specified element in this list,

**Exceptions:**

*NoSuchElementException* if value is not in the list

Implements **decaf::util::List< E >** (p. 2340).

**6.772.3.10** `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty  
() const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size()` (p. 1192) == 0.

**Returns:**

true if the size method return 0.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 186).

**6.772.3.11** `template<typename E> virtual Iterator<E>* decaf::util::StlList< E  
>::iterator () const [inline, virtual]`

Implements `decaf::lang::Iterable< E >` (p. 2152).

**6.772.3.12** `template<typename E> virtual Iterator<E>* decaf::util::StlList< E  
>::iterator () [inline, virtual]`

**Returns:**

an iterator over a set of elements of type T.

Implements `decaf::lang::Iterable< E >` (p. 2152).

**6.772.3.13** `template<typename E> virtual std::size_t decaf::util::StlList<  
E >::lastIndexOf (const E & value) throw (  
decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that `get(i) == value` or -1 if there is no such index.

**Parameters:**

*value* - element to search for

**Returns:**

the index of the last occurrence of the specified element in this list.

**Exceptions:**

*NoSuchElementException* if value is not in the list

Implements `decaf::util::List< E >` (p. 2340).

**6.772.3.14** `template<typename E> virtual ListIterator<E>* decaf::util::StlList<  
E >::listIterator (std::size_t index) const throw (  
decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,  
virtual]`

Implements `decaf::util::List< E >` (p. 2341).

**6.772.3.15** `template<typename E> virtual ListIterator<E>*  
decaf::util::StlList< E >::listIterator (std::size_t index) throw (  
decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,  
virtual]`

**Parameters:**

*index* index of first element to be returned from the list iterator (by a call to the next method).

**Returns:**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

**Exceptions:**

*IndexOutOfBoundsException* if the index is out of range (`index < 0 || index > size()` (p. 1192))

Implements `decaf::util::List< E >` (p. 2341).

**6.772.3.16** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E  
>::listIterator () const [inline, virtual]`

Implements `decaf::util::List< E >` (p. 2341).

**6.772.3.17** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E  
>::listIterator () [inline, virtual]`

**Returns:**

a list iterator over the elements in this list (in proper sequence).

Implements `decaf::util::List< E >` (p. 2342).

**6.772.3.18** `template<typename E> virtual E decaf::util::StlList<  
E >::remove (std::size_t index) throw ( de-  
cafe::lang::exceptions::UnsupportedOperationException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,  
virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

**Parameters:**

*index* - the index of the element to be removed

**Returns:**

the element previously at the specified position

**Exceptions:**

***IndexOutOfBoundsException*** - if the index is greater than size

***UnsupportedOperationException*** - If the collection is non-modifiable.

Implements **decaf::util::List**< E > (p. 2342).

```
6.772.3.19  template<typename E> virtual bool decaf::util::StlList<
               E >::remove (const E & value) throw (
               lang::exceptions::UnsupportedOperationException,
               lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an *UnsupportedOperationException* if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters:**

***value*** - element to be removed from this collection, if present

**Returns:**

true if an element was removed as a result of this call

**Exceptions:**

***UnsupportedOperationException*** if the remove operation is not supported by this collection.

***IllegalArgumentException*** If the value is not a valid entry for this **Collection** (p. 1184).

Reimplemented from **decaf::util::AbstractCollection**< E > (p. 187).

```
6.772.3.20  template<typename E> virtual E decaf::util::StlList< E
               >::set (std::size_t index, const E & element) throw (
               decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
               virtual]
```

Replaces the element at the specified position in this list with the specified element.

**Parameters:**

***index*** - index of the element to replace

***element*** - element to be stored at the specified position

**Returns:**

the element previously at the specified position



**Exceptions:**

*IndexOutOfBoundsException* - if the index is greater than size

Implements **decaf::util::List< E >** (p. 2342).

**6.772.3.21** `template<typename E> virtual std::size_t decaf::util::StlList< E >::size()  
() const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**Returns:**

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 1192).

Referenced by `decaf::util::StlList< cms::Connection * >::add()`, `decaf::util::StlList< cms::Connection * >::addAll()`, `decaf::util::StlList< cms::Connection * >::get()`, `decaf::util::StlList< cms::Connection * >::lastIndexOf()`, `decaf::util::StlList< cms::Connection * >::listIterator()`, `decaf::util::StlList< cms::Connection * >::remove()`, and `decaf::util::StlList< cms::Connection * >::set()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlList.h`

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

#include <src/main/decaf/util/StlMap.h> Inheritance diagram for `decaf::util::StlMap< K, V, COMPARATOR >`:

### Public Member Functions

- **StlMap** ()  
*Default constructor - does nothing.*
- **StlMap** (const **StlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual ~**StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const  
*Comparison, equality is dependent on the method of determining if the element are equal.*  
**Parameters:**  
*source* - **Map** (p. 2459) to compare to this one.  
**Returns:**  
*true if the **Map** (p. 2459) passed is equal in value to this one.*
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
- virtual void **copy** (const **StlMap** &source)  
*Copies the content of the source map into this map.*  
*Erases all existing data in this map.*  
**Parameters:**  
*source* The source object to copy from.
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*  
**Exceptions:**  
*UnsupportedOperationException* if this map is unmodifiable.
- virtual bool **containsKey** (const K &key) const  
*Indicates whether or this map contains a value for the given key.*

**Parameters:**

*key* The key to look up.

**Returns:**

*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const

*Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

**Parameters:**

*value* The Value to look up.

**Returns:**

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

**Returns:**

*if the **Map** (p. 2459) contains any element or not, TRUE or FALSE*

- virtual std::size\_t **size** () const

**Returns:**

*The number of elements (key/value pairs) in this map.*

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2459).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

**Parameters:**

*key* The search key.

**Returns:**

*A reference to the value for the given key.*

**Exceptions:**

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2459).*

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2459).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

**Parameters:**

*key* The search key.

**Returns:**

*A {const} reference to the value for the given key.*

**Exceptions:**

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2459).*

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

**Parameters:**

*key* The target key.  
*value* The value to be set.

**Exceptions:**

**UnsupportedOperationException** if this map is unmodifiable.

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.*

**Parameters:**

*other* A **Map** (p. 2459) instance whose elements are to all be inserted in this **Map** (p. 2459).

**Exceptions:**

**UnsupportedOperationException** If the implementing class does not support the putAll operation.

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*

**Parameters:**

*key* The search key.

**Returns:**

*a copy of the element that was previously mapped to the given key*

**Exceptions:**

**NoSuchElementException** if this key is not in the **Map** (p. 2459).  
**UnsupportedOperationException** if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3439) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2155), **Set.remove** (p. 187), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.*

**Returns:**

*the entire set of keys in this map as a std::vector.*

- virtual std::vector< V > **values** () const

**Returns:**

*the entire set of values in this map as a std::vector.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

### 6.773.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::StlMap< K, V, COMPARATOR >
```

**Map** (p. 2459) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

**Since:**

1.0

### 6.773.2 Constructor & Destructor Documentation

**6.773.2.1** `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap () [inline]`

Default constructor - does nothing.

**6.773.2.2** `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

**Parameters:**

*source* The source map.

**6.773.2.3** `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const Map< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

**Parameters:**

*source* The source map.

**6.773.2.4** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::StlMap< K, V, COMPARATOR >::~~StlMap () [inline, virtual]`

## 6.773.3 Member Function Documentation

**6.773.3.1** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear () throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all keys and values from this map.

**Exceptions:**

*UnsupportedOperationException* if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2460).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

**6.773.3.2** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

**Parameters:**

*key* The key to look up.

**Returns:**

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2461).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`.

**6.773.3.3** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR  
>::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

**Parameters:**

*value* The Value to look up.

**Returns:**

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2462).

**6.773.3.4** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR  
>::copy (const Map< K, V, COMPARATOR > & source) [inline,  
virtual]`

**6.773.3.5** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR  
>::copy (const StlMap< K, V, COMPARATOR > & source) [inline,  
virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

**Parameters:**

*source* The source object to copy from.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2463).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::StlMap()`.

**6.773.3.6** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

**6.773.3.7** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const StlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

#### Parameters:

*source* - **Map** (p. 2459) to compare to this one.

#### Returns:

true if the **Map** (p. 2459) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2463).

**6.773.3.8** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & key) const throw ( lang::exceptions::NoSuchElementException ) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2459).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

#### Parameters:

*key* The search key.

#### Returns:

A {const} reference to the value for the given key.

#### Exceptions:

**NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2459).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2463).

**6.773.3.9** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & key) throw ( lang::exceptions::NoSuchElementException ) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2459).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.



**Parameters:**

*key* The search key.

**Returns:**

A reference to the value for the given key.

**Exceptions:**

*NoSuchElementException* if the key requests doesn't exist in the **Map** (p. 2459).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2464).

**6.773.3.10** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty () const [inline, virtual]`

**Returns:**

if the **Map** (p. 2459) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2465).

**6.773.3.11** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet () const [inline, virtual]`

Returns a **Set** (p. 3439) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2155), **Set.remove** (p. 187), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

**Returns:**

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2466).

**6.773.3.12** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3701).

**6.773.3.13** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

***RuntimeException*** if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3702).

**6.773.3.14** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

***RuntimeException*** if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.773.3.15** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::put (const K & key, const V & value) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Sets the value for the specified key.

**Parameters:**

***key*** The target key.

***value*** The value to be set.

**Exceptions:**

***UnsupportedOperationException*** if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2467).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::putAll()`.

**6.773.3.16** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

**6.773.3.17** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const StlMap< K, V, COMPARATOR > & other) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2459) in this one.

**Parameters:**

*other* A **Map** (p. 2459) instance whose elements are to all be inserted in this **Map** (p. 2459).

**Exceptions:**

*UnsupportedOperationException* If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2468).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

**6.773.3.18** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

**Parameters:**

*key* The search key.

**Returns:**

a copy of the element that was previously mapped to the given key

**Exceptions:**

*NoSuchElementException* if this key is not in the **Map** (p. 2459).

*UnsupportedOperationException* if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2468).

**6.773.3.19** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size () const [inline, virtual]`

**Returns:**

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2469).

**6.773.3.20** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3704).

**6.773.3.21** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

**Exceptions:**

*RuntimeException* if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3705).

**6.773.3.22** `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values () const [inline, virtual]`

**Returns:**

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2470).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::values()`.

```

6.773.3.23  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap< K,
            V, COMPARATOR >::wait (long long millisecs, int
            nanos) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalArgumentException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters:

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE  
*nanos* additional time in nanoseconds with a range of 0-999999

#### Exceptions:

***IllegalArgumentException*** if an error occurs or the nanos argument is not in the range of [0-999999]  
***RuntimeException*** if an error occurs while waiting on the object.  
***InterruptedException*** if the wait is interrupted before it completes.  
***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

```

6.773.3.24  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap<
            K, V, COMPARATOR >::wait (long long millisecs)
            throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters:

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.  
***InterruptedException*** if the wait is interrupted before it completes.  
***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

**6.773.3.25** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR  
>::wait () throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

**Exceptions:**

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlMap.h`

## 6.774 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 3149) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.

#include <src/main/decaf/util/StlQueue.h> Inheritance diagram for decaf::util::StlQueue< T >:

### Data Structures

- class **QueueIterator**

### Public Member Functions

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > \* **iterator** ()  
*Gets an **Iterator** (p. 2154) over this **Queue** (p. 3149).*
- void **clear** ()  
*Empties this queue.*
- T & **front** ()  
*Returns a Reference to the element at the head of the queue.*
- const T & **front** () const  
*Returns a Reference to the element at the head of the queue.*
- T & **back** ()  
*Returns a Reference to the element at the tail of the queue.*
- const T & **back** () const  
*Returns a Reference to the element at the tail of the queue.*
- void **push** (const T &t)  
*Places a new Object at the Tail of the queue.*
- void **enqueueFront** (const T &t)  
*Places a new Object at the front of the queue.*
- T **pop** ()  
*Removes and returns the element that is at the Head of the queue.*
- size\_t **size** () const  
*Gets the Number of elements currently in the **Queue** (p. 3149).*
- bool **empty** () const  
*Checks if this **Queue** (p. 3149) is currently empty.*

- virtual std::vector< T > **toArray** () const
- void **reverse** (StlQueue< T > &target) const  
*Reverses the order of the contents of this queue and stores them in the target queue.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*
- T & **getSafeValue** ()  
*Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.*

### 6.774.1 Detailed Description

template<typename T> class decaf::util::StlQueue< T >

The **Queue** (p. 3149) class accepts messages with an psuh(m) command where m is the message to be queued. It destructively returns the message with **pop()** (p. 3619). **pop()** (p. 3619) returns messages in the order they were enqueued.



**Queue** (p.3149) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually returns a reference to the element popped. This frees the app from having to call the **front** method before calling pop.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues a message  
m string s = sq.pop(); // dequeues a message
```

= DESIGN CONSIDERATIONS

The **Queue** (p.3149) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p.3149).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

## 6.774.2 Constructor & Destructor Documentation

**6.774.2.1** `template<typename T> decaf::util::StlQueue< T >::StlQueue ()`  
[inline]

**6.774.2.2** `template<typename T> virtual decaf::util::StlQueue< T >::~~StlQueue ()`  
[inline, virtual]

## 6.774.3 Member Function Documentation

**6.774.3.1** `template<typename T> const T& decaf::util::StlQueue< T >::back ()`  
const [inline]

Returns a Reference to the element at the tail of the queue.

**Returns:**

reference to a queue type object or (safe)

**6.774.3.2** `template<typename T> T& decaf::util::StlQueue< T >::back ()` [inline]

Returns a Reference to the element at the tail of the queue.

**Returns:**

reference to a queue type object or (safe)

**6.774.3.3** `template<typename T> void decaf::util::StlQueue< T >::clear ()`  
[inline]

Empties this queue.

**6.774.3.4** `template<typename T> bool decaf::util::StlQueue< T >::empty () const`  
[inline]

Checks if this **Queue** (p.3149) is currently empty.

**Returns:**

boolean indicating queue emptiness

**6.774.3.5** `template<typename T> void decaf::util::StlQueue< T >::enqueueFront (const T & t) [inline]`

Places a new Object at the front of the queue.

**Parameters:**

*t* - **Queue** (p. 3149) Object Type reference.

**6.774.3.6** `template<typename T> const T& decaf::util::StlQueue< T >::front () const [inline]`

Returns a Reference to the element at the head of the queue.

**Returns:**

reference to a queue type object or (safe)

**6.774.3.7** `template<typename T> T& decaf::util::StlQueue< T >::front () [inline]`

Returns a Reference to the element at the head of the queue.

**Returns:**

reference to a queue type object or (safe)

**6.774.3.8** `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue () [inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

**Returns:**

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

**6.774.3.9** `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator () [inline]`

Gets an **Iterator** (p. 2154) over this **Queue** (p. 3149).

**Returns:**

new iterator pointer that is owned by the caller.

**6.774.3.10** `template<typename T> virtual void decaf::util::StlQueue< T >::lock ()  
throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3701).

**6.774.3.11** `template<typename T> virtual void decaf::util::StlQueue< T  
>::notify () throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3702).

**6.774.3.12** `template<typename T> virtual void decaf::util::StlQueue< T  
>::notifyAll () throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3703).

**6.774.3.13** `template<typename T> T decaf::util::StlQueue< T >::pop () [inline]`

Removes and returns the element that is at the Head of the queue.

**Returns:**

reference to a queue type object or (safe)

**6.774.3.14** `template<typename T> void decaf::util::StlQueue< T >::push (const T & t) [inline]`

Places a new Object at the Tail of the queue.

**Parameters:**

*t* - **Queue** (p. 3149) Object Type reference.

**6.774.3.15** `template<typename T> void decaf::util::StlQueue< T >::reverse (StlQueue< T > & target) const [inline]`

Reverses the order of the contents of this queue and stores them in the target queue.

**Parameters:**

*target* - The target queue that will receive the contents of this queue in reverse order.

**6.774.3.16** `template<typename T> size_t decaf::util::StlQueue< T >::size () const [inline]`

Gets the Number of elements currently in the **Queue** (p. 3149).

**Returns:**

**Queue** (p. 3149) Size

**6.774.3.17** `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray () const [inline, virtual]`

**Returns:**

the all values in this queue as a std::vector.

**6.774.3.18** `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

***RuntimeException*** if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

**6.774.3.19** `template<typename T> virtual void decaf::util::StlQueue< T >::unlock  
( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline,  
virtual]`

Unlocks the object.

#### Exceptions:

*RuntimeException* if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).

**6.774.3.20** `template<typename T> virtual void decaf::util::StlQueue<  
T >::wait (long long millisecs, int nanos) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters:

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

#### Exceptions:

*IllegalArgumentException* if an error occurs or the nanos argument is not in the range of [0-999999]

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.774.3.21** `template<typename T> virtual void decaf::util::StlQueue< T >::wait  
(long long millisecs) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters:**

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

```
6.774.3.22  template<typename T> virtual void decaf::util::StlQueue< T
>::wait () throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

**Exceptions:**

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlQueue.h

## 6.775 decaf::util::StlSet< E > Class Template Reference

**Set** (p. 3439) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

#include <src/main/decaf/util/StlSet.h> Inheritance diagram for `decaf::util::StlSet< E >`:

### Data Structures

- class **ConstSetIterator**
- class **SetIterator**

### Public Member Functions

- **StlSet** ()  
*Default constructor - does nothing.*
- **StlSet** (const **StlSet** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlSet** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual ~**StlSet** ()
- **Iterator**< E > \* **iterator** ()  
**Returns:**  
*an iterator over a set of elements of type T.*
- **Iterator**< E > \* **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const  
*Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.*  
**Returns:**  
*true if the Collections contain the same elements.*
- virtual void **copy** (const **StlSet** &source)
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).  
The collection will be empty after this method returns.  
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2155) operation. Most implementations will probably choose to override this method for efficiency.  
Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*  
**Exceptions:**  
***UnsupportedOperationException** if the clear operation is not supported by this collection*

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

*This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

**Parameters:**

*value* - the value whose presence is to be queried for in this **Collection** (p. 1184).

**Returns:**

*true if the value is contained in this collection*

**Exceptions:**

*Exception if an error occurs,*

- virtual bool **isEmpty** () const
- virtual std::size\_t **size** () const
- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Returns true if this collection changed as a result of the call.*

*(Returns false if this collection does not permit duplicates and already contains the specified element.)*

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1184) classes should clearly specify in their documentation any restrictions on what elements may be added.*

*If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.*

*For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.*

**Parameters:**

*value* - reference to the element to add.

**Returns:**

*true if the element was added*

**Exceptions:**

*UnsupportedOperationException*

*IllegalArgumentException*

*IllegalStateException if the element cannot be added at this time due to insertion restrictions*

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element *e* such that *e == o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

**Parameters:**

*value* - element to be removed from this collection, if present



**Returns:**

*true if an element was removed as a result of this call*

**Exceptions:**

***UnsupportedOperationException** if the remove operation is not supported by this collection.*

***IllegalArgumentException** If the value is not a valid entry for this **Collection** (p. 1184).*

**6.775.1 Detailed Description**

```
template<typename E> class decaf::util::StlSet< E >
```

Set (p. 3439) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

**6.775.2 Constructor & Destructor Documentation**

```
6.775.2.1 template<typename E> decaf::util::StlSet< E >::StlSet () [inline]
```

Default constructor - does nothing.

```
6.775.2.2 template<typename E> decaf::util::StlSet< E >::StlSet (const StlSet< E  
> & source) [inline]
```

Copy constructor - copies the content of the given set into this one.

**Parameters:**

*source* The source set.

```
6.775.2.3 template<typename E> decaf::util::StlSet< E >::StlSet (const  
Collection< E > & source) [inline]
```

Copy constructor - copies the content of the given set into this one.

**Parameters:**

*source* The source set.

```
6.775.2.4 template<typename E> virtual decaf::util::StlSet< E >::~~StlSet ()  
[inline, virtual]
```

**6.775.3 Member Function Documentation**

```
6.775.3.1 template<typename E> virtual bool decaf::util::StlSet< E >::add (const  
E & value) throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException,  
lang::exceptions::IllegalStateException ) [inline,  
virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1184) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters:

*value* - reference to the element to add.

#### Returns:

true if the element was added

#### Exceptions:

*UnsupportedOperationException*

*IllegalArgumentException*

*IllegalStateException* if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1186).

**6.775.3.2** `template<typename E> virtual void decaf::util::StlSet< E >::clear ()  
throw ( lang::exceptions::UnsupportedOperationException ) [inline,  
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.2155) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

#### Exceptions:

*UnsupportedOperationException* if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p.183).

**6.775.3.3** `template<typename E> virtual bool decaf::util::StlSet< E >::contains  
(const E & value) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

**Parameters:**

*value* - the value whose presence is to be queried for in this **Collection** (p.1184).

**Returns:**

true if the value is contained in this collection

**Exceptions:**

*Exception* if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p.184).

**6.775.3.4** `template<typename E> virtual void decaf::util::StlSet< E >::copy (const  
StlSet< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

**6.775.3.5** `template<typename E> virtual bool decaf::util::StlSet< E >::equals  
(const StlSet< E > & source) const [inline, virtual]`

Compares the passed collection to this one, if they contain the same elements, i.e.

all their elements are equivalent, then it returns true.

**Returns:**

true if the Collections contain the same elements.

Implements **decaf::util::Collection< E >** (p.1189).

**6.775.3.6** `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty ()  
const [inline, virtual]`

**Returns:**

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p.186).

**6.775.3.7** `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ()  
const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.2152).

**6.775.3.8** `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ()`  
`[inline, virtual]`

**Returns:**

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 2152).

**6.775.3.9** `template<typename E> virtual bool decaf::util::StlSet<`  
`E >::remove (const E & value) throw (`  
`lang::exceptions::UnsupportedOperationException,`  
`lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters:**

*value* - element to be removed from this collection, if present

**Returns:**

true if an element was removed as a result of this call

**Exceptions:**

***UnsupportedOperationException*** if the remove operation is not supported by this collection.

***IllegalArgumentException*** If the value is not a valid entry for this **Collection** (p. 1184).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 187).

**6.775.3.10** `template<typename E> virtual std::size_t decaf::util::StlSet< E >::size`  
`() const [inline, virtual]`

**Returns:**

The number of elements in this set.

Implements **decaf::util::Collection< E >** (p. 1192).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

## 6.776 activemq::wireformat::stomp::StompCommandConstants Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

### Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR\_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER\_DESTINATION**
- static const std::string **HEADER\_TRANSACTIONID**
- static const std::string **HEADER\_CONTENTLENGTH**
- static const std::string **HEADER\_SESSIONID**
- static const std::string **HEADER\_RECEIPT\_REQUIRED**
- static const std::string **HEADER\_RECEIPTID**
- static const std::string **HEADER\_MESSAGEID**
- static const std::string **HEADER\_ACK**
- static const std::string **HEADER\_LOGIN**
- static const std::string **HEADER\_PASSWORD**
- static const std::string **HEADER\_CLIENT\_ID**
- static const std::string **HEADER\_MESSAGE**
- static const std::string **HEADER\_CORRELATIONID**
- static const std::string **HEADER\_REQUESTID**
- static const std::string **HEADER\_RESPONSEID**
- static const std::string **HEADER\_EXPIRES**
- static const std::string **HEADER\_PERSISTENT**
- static const std::string **HEADER\_REPLYTO**
- static const std::string **HEADER\_TYPE**
- static const std::string **HEADER\_DISPATCH\_ASYNC**
- static const std::string **HEADER\_EXCLUSIVE**
- static const std::string **HEADER\_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER\_NOLOCAL**
- static const std::string **HEADER\_PREFETCHSIZE**
- static const std::string **HEADER\_JMSPRIORITY**
- static const std::string **HEADER\_CONSUMERPRIORITY**
- static const std::string **HEADER\_RETROACTIVE**
- static const std::string **HEADER\_SUBSCRIPTIONNAME**
- static const std::string **HEADER\_OLDSUBSCRIPTIONNAME**

- static const std::string **HEADER\_TIMESTAMP**
- static const std::string **HEADER\_REDELIVERED**
- static const std::string **HEADER\_REDELIVERYCOUNT**
- static const std::string **HEADER\_SELECTOR**
- static const std::string **HEADER\_ID**
- static const std::string **HEADER\_SUBSCRIPTION**
- static const std::string **HEADER\_TRANSFORMATION**
- static const std::string **HEADER\_TRANSFORMATION\_ERROR**
- static const std::string **ACK\_CLIENT**
- static const std::string **ACK\_AUTO**
- static const std::string **ACK\_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE\_PREFIX**
- static const std::string **TOPIC\_PREFIX**
- static const std::string **TEMPQUEUE\_PREFIX**
- static const std::string **TEMPTOPIC\_PREFIX**

## 6.776.1 Field Documentation

- 6.776.1.1** `const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT`  
[static]
- 6.776.1.2** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK`  
[static]
- 6.776.1.3** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO`  
[static]
- 6.776.1.4** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT` [static]
- 6.776.1.5** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL` [static]
- 6.776.1.6** `const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN`  
[static]
- 6.776.1.7** `const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES`  
[static]
- 6.776.1.8** `const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT`  
[static]
- 6.776.1.9** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT`  
[static]
- 6.776.1.10** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED`  
[static]
- 6.776.1.11** `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT`  
[static]
- 6.776.1.12** `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.776.1.13** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.776.1.14** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
- 6.776.1.15** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`



## 6.777 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

### Public Member Functions

- **StompFrame** ()  
*Default constructor.*
- virtual **~StompFrame** ()  
*Destruction.*
- **StompFrame \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- void **copy** (const **StompFrame** \*src)  
*Copies the contents of the passed Frame to this one.*
- void **setCommand** (const std::string &cmd)  
*Sets the command for this **stomp** (p. 139) frame.*
- const std::string & **getCommand** () const  
*Accessor for this frame's command field.*
- bool **hasProperty** (const std::string &name) const  
*Checks if the given property is present in the Frame.*
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const  
*Gets a property from this Frame's properties and returns it, or the default value given.*
- std::string **removeProperty** (const std::string &name)  
*Gets and remove the property specified, if the property is not set, this method returns the empty string.*
- void **setProperty** (const std::string &name, const std::string &value)  
*Sets the property given to the value specified in this Frame's Properties.*
- **decaf::util::Properties** & **getProperties** ()  
*Gets access to the header properties for this frame.*
- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const  
*Accessor for the body data of this frame.*
- std::vector< unsigned char > & **getBody** ()  
*Non-const version of the body accessor.*

- `std::size_t getBodyLength () const`  
*Return the number of bytes contained in this frames body.*
- `void setBody (const unsigned char *bytes, std::size_t numBytes)`  
*Sets the body data of this frame as a byte sequence.*
- `void toStream (decaf::io::DataOutputStream *stream) const throw ( decaf::io::IOException )`  
*Writes this Frame to an OuputStream in the Stomp Wire Format.*
- `void fromStream (decaf::io::DataInputStream *stream) throw ( decaf::io::IOException )`  
*Reads a Stop Frame from a DataInputStream in the Stomp Wire format.*

### 6.777.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

### 6.777.2 Constructor & Destructor Documentation

#### 6.777.2.1 `activemq::wireformat::stomp::StompFrame::StompFrame () [inline]`

Default constructor.

#### 6.777.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame () [inline, virtual]`

Destruction.

### 6.777.3 Member Function Documentation

#### 6.777.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone () const`

Clone this message exactly, returns a new instance that the caller is required to delete.

**Returns:**

new copy of this message

#### 6.777.3.2 `void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * src)`

Copies the contents of the passed Frame to this one.

**Parameters:**

*src* - Frame to copy

**6.777.3.3** void activemq::wireformat::stomp::StompFrame::fromStream  
(decaf::io::DataInputStream \* *stream*) throw ( decaf::io::IOException )

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

**Parameters:**

*stream* - The stream to read the Frame from.

**Exceptions:**

*IOException* if an error occurs while writing the Frame.

**6.777.3.4** std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ()  
[inline]

Non-const version of the body accessor.

**6.777.3.5** const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const  
[inline]

Accessor for the body data of this frame.

**Returns:**

char pointer to body data

**6.777.3.6** std::size\_t activemq::wireformat::stomp::StompFrame::getBodyLength ()  
const [inline]

Return the number of bytes contained in this frames body.

**Returns:**

Body bytes length.

**6.777.3.7** const std::string& activemq::wireformat::stomp::StompFrame::getCommand ()  
const [inline]

Accessor for this frame's command field.

**6.777.3.8** const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ()  
const [inline]

**6.777.3.9** decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ()  
[inline]

Gets access to the header properties for this frame.

**Returns:**

the Properties object owned by this Frame

**6.777.3.10** `std::string activemq::wireformat::stomp::StompFrame::getProperty  
(const std::string & name, const std::string & fallback = "") const` `[inline]`

Gets a property from this Frame's properties and returns it, or the default value given.

**Parameters:**

*name* - The name of the property to lookup

*fallback* - The default value to return if this value isn't set

**Returns:**

string value of the property asked for.

**6.777.3.11** `bool activemq::wireformat::stomp::StompFrame::hasProperty (const  
std::string & name) const` `[inline]`

Checks if the given property is present in the Frame.

**Parameters:**

*name* - The name of the property to check for.

**6.777.3.12** `std::string activemq::wireformat::stomp::StompFrame::removeProperty  
(const std::string & name)` `[inline]`

Gets and remove the property specified, if the property is not set, this method returns the empty string.

**Parameters:**

*name* - the Name of the property to get and return.

**6.777.3.13** `void activemq::wireformat::stomp::StompFrame::setBody (const  
unsigned char * bytes, std::size_t numBytes)`

Sets the body data of this frame as a byte sequence.

**Parameters:**

*bytes* The byte buffer to be set in the body.

*numBytes* The number of bytes in the buffer.

**6.777.3.14** void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & *cmd*) [inline]

Sets the command for this **stomp** (p. 139) frame.

**Parameters:**

*cmd* command The command to be set.

**6.777.3.15** void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & *name*, const std::string & *value*) [inline]

Sets the property given to the value specified in this Frame's Properties.

**Parameters:**

*name* - Name of the property.

*value* - Value to set the property to.

**6.777.3.16** void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream \* *stream*) const throw ( decaf::io::IOException )

Writes this Frame to an OuputStream in the Stomp Wire Format.

**Parameters:**

*stream* - The stream to write the Frame to.

**Exceptions:**

*IOException* if an error occurs while reading the Frame.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompFrame.h**

## 6.778 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

### Public Member Functions

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)
 

*Converts the Headers in a Stomp Frame into Headers in the given Message Command.*
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)
 

*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3633).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)
 

*Converts from a Stomp Destination to an ActiveMQDestination.*
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
 

*Converts from a ActiveMQDestination to a Stomp Destination Name.*
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)
 

*Converts a MessageId instance to a Stomp MessageId String.*
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)
 

*Converts a Stomp MessageId string to a MessageId.*
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
 

*Converts a ConsumerId instance to a Stomp ConsumerId String.*
- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)
 

*Converts a Stomp ConsumerId string to a ConsumerId.*
- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)
 

*Converts a ProducerId instance to a Stomp ProducerId String.*
- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)
 

*Converts a Stomp ProducerId string to a ProducerId.*
- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
 

*Converts a TransactionId instance to a Stomp TransactionId String.*
- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)
 

*Converts a Stomp TransactionId string to a TransactionId.*

## 6.778.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since:

3.0

## 6.778.2 Constructor & Destructor Documentation

**6.778.2.1** `activemq::wireformat::stomp::StompHelper::StompHelper () [inline]`

**6.778.2.2** `virtual activemq::wireformat::stomp::StompHelper::~~StompHelper () [inline, virtual]`

## 6.778.3 Member Function Documentation

**6.778.3.1** `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & consumerId)`

Converts a Stomp ConsumerId string to a ConsumerId.

**Parameters:**

*consumerId* - the String Consumer Id to convert.

**Returns:**

Pointer to a new ConsumerId.

**6.778.3.2** `std::string activemq::wireformat::stomp::StompHelper::convertConsumerId (const Pointer< ConsumerId > & consumerId)`

Converts a ConsumerId instance to a Stomp ConsumerId String.

**Parameters:**

*consumerId* - the Consumer instance to convert.

**Returns:**

a Stomp Consumer Id String.

**6.778.3.3** `std::string activemq::wireformat::stomp::StompHelper::convertDestination (const Pointer< ActiveMQDestination > & destination)`

Converts from a ActiveMQDestination to a Stomp Destination Name.

**Parameters:**

*destination* - The ActiveMQDestination to Convert

**Returns:**

the Stomp String name that defines the destination.

**6.778.3.4** `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & destination)`

Converts from a Stomp Destination to an ActiveMQDestination.

**Parameters:**

*destination* - The Stomp Destination name string.

**Returns:**

Pointer to a new ActiveMQDestination.

**6.778.3.5** `Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId (const std::string & messageId)`

Converts a Stomp MessageId string to a MessageId.

**Parameters:**

*messageId* - the String message Id to convert.

**Returns:**

Pointer to a new MessageId.

**6.778.3.6** `std::string activemq::wireformat::stomp::StompHelper::convertMessageId (const Pointer< MessageId > & messageId)`

Converts a MessageId instance to a Stomp MessageId String.

**Parameters:**

*messageId* - the MessageId instance to convert.

**Returns:**

a Stomp Message Id String.

**6.778.3.7** `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & producerId)`

Converts a Stomp ProducerId string to a ProducerId.



**Parameters:**

*producerId* - the String Producer Id to convert.

**Returns:**

Pointer to a new ProducerId.

### 6.778.3.8 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & producerId)`

Converts a ProducerId instance to a Stomp ProducerId String.

**Parameters:**

*producerId* - the Producer instance to convert.

**Returns:**

a Stomp Producer Id String.

### 6.778.3.9 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & message, const Pointer< StompFrame > & frame)`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3633).

**Parameters:**

*message* - The message to move the Headers to.

*frame* - The frame to extract headers from.

### 6.778.3.10 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & frame, const Pointer< Message > & message)`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

**Parameters:**

*frame* - The frame to extract headers from.

*message* - The message to move the Headers to.

### 6.778.3.11 `Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & transactionId)`

Converts a Stomp TransactionId string to a TransactionId.

**Parameters:**

*transactionId* - the String Transaction Id to convert.

**Returns:**

Pointer to a new TransactionId.

**6.778.3.12** `std::string activemq::wireformat::stomp::StompHelper::convertTransactionId (const Pointer< TransactionId > & transactionId)`

Converts a TransactionId instance to a Stomp TransactionId String.

**Parameters:**

*transactionId* - the Transaction instance to convert.

**Returns:**

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompHelper.h`

## 6.779 activemq::wireformat::stomp::StompWireFormat Class Reference

#include <src/main/activemq/wireformat/stomp/StompWireFormat.h> Inheritance diagram for activemq::wireformat::stomp::StompWireFormat:

### Public Member Functions

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( **decaf::io::IOException** )

*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )

*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*

- virtual void **setVersion** (int version AMQCPP\_UNUSED)

*Set the Version.*

- virtual int **getVersion** () const

*Get the Version.*

- virtual bool **inReceive** () const

*Is there a Message being unmarshaled?*

- virtual bool **hasNegotiator** () const

*Returns true if this **WireFormat** (p. 3976) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw ( **decaf::lang::exceptions::UnsupportedOperationException** )

*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*

## 6.779.1 Constructor & Destructor Documentation

**6.779.1.1** `activemq::wireformat::stomp::StompWireFormat::StompWireFormat ()`

**6.779.1.2** `virtual  
activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat ()  
[virtual]`

## 6.779.2 Member Function Documentation

**6.779.2.1** `virtual Pointer<transport::Transport> ac-  
tivemq::wireformat::stomp::StompWireFormat::createNegotiator  
(const Pointer< transport::Transport > & transport) throw (  
decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

### Returns:

new instance of a **WireFormatNegotiator** (p. 4016).

### Exceptions:

*UnsupportedOperationException* if the **WireFormat** (p. 3976) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3977).

**6.779.2.2** `virtual int activemq::wireformat::stomp::StompWireFormat::getVersion  
() const [inline, virtual]`

Get the Version.

### Returns:

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3977).

**6.779.2.3** `virtual bool ac-  
tivemq::wireformat::stomp::StompWireFormat::hasNegotiator () const  
[inline, virtual]`

Returns true if this **WireFormat** (p. 3976) has a Negotiator that needs to wrap the Transport that uses it.

### Returns:

true if the **WireFormat** (p. 3976) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3977).

#### 6.779.2.4 virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive ( ) const [inline, virtual]

Is there a Message being unmarshaled?

##### Returns:

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3978).

#### 6.779.2.5 virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport \* *transport*, decaf::io::DataOutputStream \* *out*) throw ( decaf::io::IOException ) [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

##### Parameters:

*command* The Command to Marshal to the output stream.

*transport* (p. 97) The Transport that initiated this marshal call.

*out* The output stream to write the command to.

##### Exceptions:

*IOException*

Implements **activemq::wireformat::WireFormat** (p. 3978).

#### 6.779.2.6 virtual void activemq::wireformat::stomp::StompWireFormat::setVersion (int version *AMQCPP\_UNUSED*) [inline, virtual]

Set the Version.

##### Parameters:

*the* version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3978).

#### 6.779.2.7 virtual Pointer<commands::Command> ac- tivemq::wireformat::stomp::StompWireFormat::unmarshal (const activemq::transport::Transport \* *transport*, decaf::io::DataInputStream \* *in*) throw ( decaf::io::IOException ) [virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

##### Parameters:

*transport* (p. 97) - Pointer to the **transport** (p. 97) that is making this request.

*in* - the input stream to read the command from.

**Returns:**

the newly marshaled Command, caller owns the pointer

**Exceptions:**

*IOException*

Implements **activemq::wireformat::WireFormat** (p. 3979).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormat.h`

## 6.780 activemq::wireformat::stomp::StompWireFormatFactory Class Reference

Factory used to create the Stomp Wire Format instance.

#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h> Inheritance diagram for activemq::wireformat::stomp::StompWireFormatFactory:

### Public Member Functions

- **StompWireFormatFactory** ()
- virtual **~StompWireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::IllegalStateException** )  
*Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.780.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

### 6.780.2 Constructor & Destructor Documentation

**6.780.2.1** **activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory** () [inline]

**6.780.2.2** **virtual**  
**activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory** () [inline, virtual]

### 6.780.3 Member Function Documentation

**6.780.3.1** **virtual Pointer<WireFormat> ac-**  
**tivemq::wireformat::stomp::StompWireFormatFactory::createWireFormat**  
 (const **decaf::util::Properties** & *properties*) throw ( **decaf::lang::exceptions::IllegalStateException** ) [virtual]

Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters:

*properties* - the Properties for this **WireFormat** (p. 3976)

Implements **activemq::wireformat::WireFormatFactory** (p. 3980).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormatFactory.h**

## 6.781 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

#include <src/main/cms/Stoppable.h> Inheritance diagram for cms::Stoppable:

### Public Member Functions

- virtual **~Stoppable** ()
- virtual void **stop** ()=0 throw ( CMSEException )  
*Stops this service.*

### 6.781.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since:

1.0

### 6.781.2 Constructor & Destructor Documentation

**6.781.2.1** virtual cms::Stoppable::~~Stoppable () [inline, virtual]

### 6.781.3 Member Function Documentation

**6.781.3.1** virtual void cms::Stoppable::stop () throw ( CMSEException ) [pure virtual]

Stops this service.

Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs while stopping the Service.

Implemented in **activemq::core::ActiveMQConnection** (p. 290).

The documentation for this class was generated from the following file:

- src/main/cms/**Stoppable.h**



## 6.782 decaf::util::logging::StreamHandler Class Reference

Stream based **logging** (p. 175) **Handler** (p. 1978).

#include <src/main/decaf/util/logging/StreamHandler.h> Inheritance diagram for decaf::util::logging::StreamHandler:

### Public Member Functions

- **StreamHandler** ()  
*Create a **StreamHandler** (p. 3649), with no current output stream.*
- **StreamHandler** (decaf::io::OutputStream \*stream, Formatter \*formatter)  
*Create a **StreamHandler** (p. 3649), with no current output stream.*
- virtual ~**StreamHandler** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Close the current output stream.*
- virtual void **flush** ()  
*Flush the Handler's output, clears any buffers.*
- virtual void **publish** (const **LogRecord** &record)  
*Publish the Log Record to this **Handler** (p. 1978).*
- virtual bool **isLoggable** (const **LogRecord** &record) const  
*Check if this **Handler** (p. 1978) would actually log a given **LogRecord** (p. 2413).*

### Protected Member Functions

- virtual void **setOutputStream** (decaf::io::OutputStream \*stream) throw ( decaf::lang::exceptions::NullPointerException )  
*Change the output stream.*
- void **close** (bool closeStream)  
*Closes this handler, but the underlying output stream is only closed if closeStream is true.*

#### 6.782.1 Detailed Description

Stream based **logging** (p. 175) **Handler** (p. 1978). This is primarily intended as a base class or support class to be used in implementing other **logging** (p. 175) Handlers.

LogRecords are published to a given decaf::io::OutputStream (p. 2907).

Configuration: By default each **StreamHandler** (p. 3649) is initialized using the following **Log-Manager** (p. 2406) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

\* `decaf.util.logging.StreamHandler.level` specifies the default level for the **Handler** (p.1978) (defaults to **Level.INFO** (p.2336)). \* `decaf.util.logging.StreamHandler.filter` specifies the name of a **Filter** (p.1893) class to use (defaults to no **Filter** (p.1893)). \* `decaf.util.logging.StreamHandler.formatter` specifies the name of a **Formatter** (p.1964) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p.3500)).

#### Since:

1.0

## 6.782.2 Constructor & Destructor Documentation

### 6.782.2.1 `decaf::util::logging::StreamHandler::StreamHandler ()`

Create a **StreamHandler** (p.3649), with no current output stream.

### 6.782.2.2 `decaf::util::logging::StreamHandler::StreamHandler (decaf::io::OutputStream * stream, Formatter * formatter)`

Create a **StreamHandler** (p.3649), with no current output stream.

### 6.782.2.3 `virtual decaf::util::logging::StreamHandler::~~StreamHandler ()` [virtual]

## 6.782.3 Member Function Documentation

### 6.782.3.1 `void decaf::util::logging::StreamHandler::close (bool closeStream)` [protected]

Closes this handler, but the underlying output stream is only closed if `closeStream` is true.

#### Parameters:

*closeStream* whether to close the underlying output stream.

### 6.782.3.2 `virtual void decaf::util::logging::StreamHandler::close () throw (decaf::io::IOException )` [virtual]

Close the current output stream. The close method will perform a flush and then close the **Handler** (p.1978). After close has been called this **Handler** (p.1978) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

#### Exceptions:

*IOException* if an I/O error occurs.

Implements **decaf::io::Closeable** (p.1149).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p.1399).

**6.782.3.3 virtual void decaf::util::logging::StreamHandler::flush () [virtual]**

Flush the Handler's output, clears any buffers.

Implements **decaf::util::logging::Handler** (p.1979).

**6.782.3.4 virtual bool decaf::util::logging::StreamHandler::isLoggable (const LogRecord & *record*) const [virtual]**

Check if this **Handler** (p.1978) would actually log a given **LogRecord** (p.2413).

**Parameters:**

*record* LogRecord (p.2413) to check

**Returns:**

true if the record can be logged with current settings.

Reimplemented from **decaf::util::logging::Handler** (p.1980).

**6.782.3.5 virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & *record*) [virtual]**

Publish the Log Record to this **Handler** (p.1978).

**Parameters:**

*record* The LogRecord (p.2413) to Publish

Implements **decaf::util::logging::Handler** (p.1980).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p.1400).

**6.782.3.6 virtual void decaf::util::logging::StreamHandler::setOutputStream (decaf::io::OutputStream \* *stream*) throw ( decaf::lang::exceptions::NullPointerException ) [protected, virtual]**

Change the output stream. If there is a current output stream then the Formatter's tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

**Parameters:**

*stream* The new output stream. May not be NULL.

**Exceptions:**

*NullPointerException* if the passed stream is NULL.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

## 6.783 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 3652).

#include <src/main/cms/StreamMessage.h> Inheritance diagram for cms::StreamMessage:

### Public Member Functions

- virtual **~StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the Stream message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the Stream message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the Stream message stream.*
- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Char from the Stream message stream.*

- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a char to the Stream message stream as a 1-byte value.*

- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit float from the Stream message stream.*

- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a float to the Stream message stream as a 4 byte value.*

- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Stream message stream.*

- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the Stream message stream as a 8 byte value.*

- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Stream message stream.*

- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the Stream message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Stream message stream.*

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the Stream message stream as a 2 byte value.*

- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Stream message stream.*

- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the Stream message stream as a 4 byte value.*

- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit long from the Stream message stream.*

- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a long long to the Stream message stream as a 8 byte value.*

- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an ASCII String from the Stream message stream.*

- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an ASCII String to the Stream message stream.*

### 6.783.1 Detailed Description

Interface for a **StreamMessage** (p. 3652). The stream Messages provides a **Message** (p. 2534) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

**StreamMessage** (p. 3652) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1160). The string-to-primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X		X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.3

## 6.783.2 Constructor & Destructor Documentation

**6.783.2.1** virtual cms::StreamMessage::~StreamMessage () [inline, virtual]

## 6.783.3 Member Function Documentation

**6.783.3.1** virtual bool cms::StreamMessage::readBoolean () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a Boolean from the Stream message stream.

### Returns:

boolean value from stream

### Exceptions:

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 541).

**6.783.3.2** virtual unsigned char cms::StreamMessage::readByte () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a Byte from the Stream message stream.

### Returns:

unsigned char value from stream

### Exceptions:

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 542).

**6.783.3.3** `virtual int cms::StreamMessage::readBytes (unsigned char *  
buffer, int length) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [pure virtual]`

Reads a portion of the Stream message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 1160) is thrown. No bytes will be read from the stream for this exception case.

**Parameters:**

*buffer* the buffer into which the data is read

*length* the number of bytes to read; must be less than or equal to value.length

**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

**CMSException** (p. 1160) - if the CMS provider fails to read the message due to some internal error.

**MessageEOFException** (p. 2661) - if unexpected end of message stream has been reached.

**MessageFormatException** (p. 2662) - if this type conversion is invalid.

**MessageNotReadableException** (p. 2723) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 542).

**6.783.3.4** `virtual int cms::StreamMessage::readBytes (std::vector< unsigned  
char > & value) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [pure virtual]`

Reads a byte array from the Stream message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

**Parameters:**

*value* buffer to place data in



**Returns:**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 543).

**6.783.3.5** `virtual char cms::StreamMessage::readChar () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )` [pure virtual]

Reads a Char from the Stream message stream.

**Returns:**

char value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 543).

**6.783.3.6** `virtual double cms::StreamMessage::readDouble () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )` [pure virtual]

Reads a 64 bit double from the Stream message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 544).

**6.783.3.7** `virtual float cms::StreamMessage::readFloat () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )` [pure virtual]

Reads a 32 bit float from the Stream message stream.

**Returns:**

double value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 544).

**6.783.3.8** `virtual int cms::StreamMessage::readInt () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )` [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

**Returns:**

int value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 545).

**6.783.3.9** `virtual long long cms::StreamMessage::readLong () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 64 bit long from the Stream message stream.

**Returns:**

long long value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 545).

**6.783.3.10** `virtual short cms::StreamMessage::readShort () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 16 bit signed short from the Stream message stream.

**Returns:**

short value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 545).

**6.783.3.11** `virtual std::string cms::StreamMessage::readString () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads an ASCII String from the Stream message stream.

**Returns:**

String from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 546).

**6.783.3.12** `virtual unsigned short cms::StreamMessage::readUnsignedShort () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

**Returns:**

unsigned short value from stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to read the message due to some internal error.

*MessageEOFException* (p. 2661) - if unexpected end of message stream has been reached.

*MessageFormatException* (p. 2662) - if this type conversion is invalid.

*MessageNotReadableException* (p. 2723) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 546).

**6.783.3.13** `virtual void cms::StreamMessage::writeBoolean (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a boolean to the Stream message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters:**

*value* boolean to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 547).

**6.783.3.14** `virtual void cms::StreamMessage::writeByte (unsigned char value)  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[pure virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

**Parameters:**

*value* byte to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 547).

**6.783.3.15** `virtual void cms::StreamMessage::writeBytes (const  
unsigned char * value, int offset, int length) throw (  
cms::MessageNotWriteableException, cms::CMSException ) [pure  
virtual]`

Writes a portion of a byte array to the Stream message stream. size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

*offset* the initial offset within the byte array

*length* the number of bytes to use

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.16** `virtual void cms::StreamMessage::writeBytes (const std::vector<  
unsigned char > & value) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [pure virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

**Parameters:**

*value* bytes to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.17** `virtual void cms::StreamMessage::writeChar (char value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a char to the Stream message stream as a 1-byte value.

**Parameters:**

*value* char to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.18** `virtual void cms::StreamMessage::writeDouble (double value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a double to the Stream message stream as a 8 byte value.

**Parameters:**

*value* double to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 549).

**6.783.3.19** `virtual void cms::StreamMessage::writeFloat (float value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a float to the Stream message stream as a 4 byte value.

**Parameters:**

*value* float to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 549).

**6.783.3.20** `virtual void cms::StreamMessage::writeInt (int value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a signed int to the Stream message stream as a 4 byte value.

**Parameters:**

*value* signed int to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 549).

**6.783.3.21** `virtual void cms::StreamMessage::writeLong (long long value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

**Parameters:**

*value* signed long long to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 550).

**6.783.3.22** `virtual void cms::StreamMessage::writeShort (short value) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

**Parameters:**

*value* signed short to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 550).

**6.783.3.23** `virtual void cms::StreamMessage::writeString (const std::string & value)  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[pure virtual]`

Writes an ASCII String to the Stream message stream.

**Parameters:**

*value* String to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 550).

**6.783.3.24** `virtual void cms::StreamMessage::writeUnsignedShort (unsigned  
short value) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [pure virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

**Parameters:**

*value* unsigned short to write to the stream

**Exceptions:**

*CMSException* (p. 1160) - if the CMS provider fails to write the message due to some internal error.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 551).

The documentation for this class was generated from the following file:

- `src/main/cms/StreamMessage.h`



## 6.784 decaf::lang::String Class Reference

The **String** (p. 3665) class represents an immutable sequence of chars.

#include <src/main/decaf/lang/String.h> Inheritance diagram for decaf::lang::String:

### Public Member Functions

- **String** ()  
*Creates a new empty **String** (p. 3665) object.*
- **String** (const std::string &source)  
*Create a new **String** (p. 3665) object that represents the given STL string.*
- virtual ~**String** ()
- bool **isEmpty** () const
- virtual int **length** () const  
**Returns:**  
*the length of the underlying character sequence.*
- virtual char **charAt** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )  
*Returns the Char at the specified index so long as the index is not greater than the length of the sequence.*  
**Parameters:**  
*index - position to return the char at.*  
**Returns:**  
*the char at the given position*  
**Exceptions:**  
*IndexOutOfBoundsException if index is > than **length**() (p. 1136) or negative*
- virtual **CharSequence** \* **subSequence** (int start, int end) const throw ( lang::exceptions::IndexOutOfBoundsException )  
*Returns a new **CharSequence** (p. 1135) that is a subsequence of this sequence.  
The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.*  
**Parameters:**  
*start - the start index, inclusive  
end - the end index, exclusive*  
**Returns:**  
*a new **CharSequence** (p. 1135)*  
**Exceptions:**  
*IndexOutOfBoundsException if start or end > **length**() (p. 1136) or start or end are negative.*
- virtual std::string **toString** () const

**Returns:**

*the string representation of this **CharSequence** (p. 1135)*

### 6.784.1 Detailed Description

The **String** (p. 3665) class represents an immutable sequence of chars.

**Since:**

1.0

### 6.784.2 Constructor & Destructor Documentation

#### 6.784.2.1 **decaf::lang::String::String ()**

Creates a new empty **String** (p. 3665) object.

#### 6.784.2.2 **decaf::lang::String::String (const std::string & *source*)**

Create a new **String** (p. 3665) object that represents the given STL string.

**Parameters:**

*source* The string to copy into this new **String** (p. 3665) object.

#### 6.784.2.3 **virtual decaf::lang::String::~~String ()** [virtual]

### 6.784.3 Member Function Documentation

#### 6.784.3.1 **virtual char decaf::lang::String::charAt (int *index*) const throw (lang::exceptions::IndexOutOfBoundsException)** [virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

**Parameters:**

*index* - position to return the char at.

**Returns:**

the char at the given position

**Exceptions:**

*IndexOutOfBoundsException* if index is > than **length()** (p. 1136) or negative

Implements **decaf::lang::CharSequence** (p. 1135).

### 6.784.3.2 bool decaf::lang::String::isEmpty () const

**Returns:**

true if the length of this **String** (p. 3665) is zero.

### 6.784.3.3 virtual int decaf::lang::String::length () const [virtual]

**Returns:**

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 1136).

### 6.784.3.4 virtual CharSequence\* decaf::lang::String::subSequence (int *start*, int *end*) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Returns a new **CharSequence** (p. 1135) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index *end* - 1. The length (in chars) of the returned sequence is *end* - *start*, so if *start* == *end* then an empty sequence is returned.

**Parameters:**

*start* - the start index, inclusive

*end* - the end index, exclusive

**Returns:**

a new **CharSequence** (p. 1135)

**Exceptions:**

*IndexOutOfBoundsException* if *start* or *end* > **length()** (p. 1136) or *start* or *end* are negative.

Implements **decaf::lang::CharSequence** (p. 1136).

### 6.784.3.5 virtual std::string decaf::lang::String::toString () const [virtual]

**Returns:**

the string representation of this **CharSequence** (p. 1135)

Implements **decaf::lang::CharSequence** (p. 1136).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**String.h**

## 6.785 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

### Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)  
*Constructs a string tokenizer for the specified string.*
- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const  
*Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.*
- virtual bool **hasMoreTokens** () const  
*Tests if there are more tokens available from this tokenizer's string.*
- virtual std::string **nextToken** () throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token from this string tokenizer.*
- virtual std::string **nextToken** (const std::string &delim) throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token in this string tokenizer's string.*
- virtual unsigned int **toArray** (std::vector< std::string > &array)  
*Grab all remaining tokens in the String and return them in the vector that is passed in by reference.*
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)  
*Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.*

### 6.785.1 Constructor & Destructor Documentation

#### 6.785.1.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false)

Constructs a string tokenizer for the specified string. All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 3668) may result in an Exception.

#### Parameters:

*str* - The string to tokenize

*delim* - String containing the delimiters

*returnDelims* - boolean indicating if the delimiters are returned as tokens

**6.785.1.2** virtual decaf::util::StringTokenizer::~~StringTokenizer () [virtual]

## 6.785.2 Member Function Documentation

**6.785.2.1** virtual int decaf::util::StringTokenizer::countTokens () const [virtual]

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception. The current position is not advanced.

### Returns:

Count of remaining tokens

**6.785.2.2** virtual bool decaf::util::StringTokenizer::hasMoreTokens () const [virtual]

Tests if there are more tokens available from this tokenizer's string.

### Returns:

true if there are more tokens remaining

**6.785.2.3** virtual std::string decaf::util::StringTokenizer::nextToken (const std::string & *delim*) throw ( lang::exceptions::NoSuchElementException ) [virtual]

Returns the next token in this string tokenizer's string. First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 3668) object is changed to be the characters in the string *delim*. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

### Parameters:

*delim* - string containing the new set of delimiters

### Returns:

next string in the token list

### Exceptions:

*NoSuchElementException*

**6.785.2.4** virtual std::string decaf::util::StringTokenizer::nextToken () throw ( lang::exceptions::NoSuchElementException ) [virtual]

Returns the next token from this string tokenizer.

**Returns:**

string value of next token

**Exceptions:**

*NoSuchElementException*

**6.785.2.5** `virtual void decaf::util::StringTokenizer::reset (const std::string & str = "", const std::string & delim = "", bool returnDelims = false) [virtual]`

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning. This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenizer the string. If set to "", no change is made If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

**Parameters:**

*str* - New String to tokenize or "", defaults to ""

*delim* - New Delimiter String to use or "", defaults to ""

*returnDelims* - Should the Tokenizer return delimiters as Tokens, default false

**6.785.2.6** `virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector< std::string > & array) [virtual]`

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

**Parameters:**

*array* - vector to place token strings in

**Returns:**

number of string placed into the vector

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StringTokenizer.h**

## 6.786 activemq::commands::SubscriptionInfo Class Reference

#include <src/main/activemq/commands/SubscriptionInfo.h> Inheritance diagram for activemq::commands::SubscriptionInfo:

### Public Member Functions

- **SubscriptionInfo** ()
- virtual **~SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SubscriptionInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &subscribedDestination)

### Static Public Attributes

- static const unsigned char **ID\_SUBSCRIPTIONINFO** = 55

## Protected Attributes

- `std::string clientId`
- `Pointer< ActiveMQDestination > destination`
- `std::string selector`
- `std::string subscriptionName`
- `Pointer< ActiveMQDestination > subscribedDestination`

## 6.786.1 Constructor & Destructor Documentation

**6.786.1.1** `activemq::commands::SubscriptionInfo::SubscriptionInfo ()`

**6.786.1.2** `virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo ()`  
[virtual]

## 6.786.2 Member Function Documentation

**6.786.2.1** `virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.786.2.2** `virtual void activemq::commands::SubscriptionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

**6.786.2.3** `virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if DataStructure's are Equal.



**6.786.2.4** virtual std::string& activemq::commands::SubscriptionInfo::getClientId () [virtual]

**6.786.2.5** virtual const std::string& activemq::commands::SubscriptionInfo::getClientId () const [virtual]

**6.786.2.6** virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns:

new **DataStructure** (p. 1660) type copy.

Implements **activemq::commands::DataStructure** (p. 1662).

- 6.786.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ()`  
[virtual]
- 6.786.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () const`  
[virtual]
- 6.786.2.9 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ()`  
[virtual]
- 6.786.2.10 `virtual const std::string& activemq::commands::SubscriptionInfo::getSelector () const`  
[virtual]
- 6.786.2.11 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ()`  
[virtual]
- 6.786.2.12 `virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName () const`  
[virtual]
- 6.786.2.13 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ()`  
[virtual]
- 6.786.2.14 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const` [virtual]
- 6.786.2.15 `virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & clientId)` [virtual]
- 6.786.2.16 `virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.786.2.17 `virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & selector)` [virtual]
- 6.786.2.18 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName (const std::string & subscriptionName)` [virtual]
- 6.786.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & subscribedDestination)` [virtual]
- 6.786.2.20 `virtual std::string activemq::commands::SubscriptionInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 833).

**6.786.3 Field Documentation**

**6.786.3.1** `std::string activemq::commands::SubscriptionInfo::clientId` [protected]

**6.786.3.2** `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination`  
[protected]

**6.786.3.3** `const unsigned char activemq::commands::SubscriptionInfo::ID__ - SUBSCRIPTIONINFO = 55` [static]

**6.786.3.4** `std::string activemq::commands::SubscriptionInfo::selector` [protected]

**6.786.3.5** `std::string activemq::commands::SubscriptionInfo::subscriptionName`  
[protected]

**6.786.3.6** `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

## 6.787 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3676).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.787.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3676).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.787.2 Constructor & Destructor Documentation

**6.787.2.1** `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

**6.787.2.2** `virtual activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

## 6.787.3 Member Function Documentation

**6.787.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.787.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.787.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.787.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.787.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.787.3.6** virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.787.3.7** virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h

## 6.788 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3680).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.788.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3680).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.788.2 Constructor & Destructor Documentation

**6.788.2.1** `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

**6.788.2.2** `virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

## 6.788.3 Member Function Documentation

**6.788.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.788.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.788.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.788.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.788.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.788.3.6** virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.788.3.7** virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**

## 6.789 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3684).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.789.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3684).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.789.2 Constructor & Destructor Documentation

**6.789.2.1** `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

**6.789.2.2** `virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

## 6.789.3 Member Function Documentation

**6.789.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.789.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.789.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.789.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - **BinaryReader** that provides that data source

**Exceptions:**

**IOException** if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.789.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - **BooleanStream** stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

**IOException** if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.789.3.6** virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.789.3.7** virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SubscriptionInfoMarshaller.h**

## 6.790 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3688).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.790.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3688).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.790.2 Constructor & Destructor Documentation

**6.790.2.1** `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

**6.790.2.2** `virtual activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

## 6.790.3 Member Function Documentation

**6.790.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.790.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.790.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.790.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.790.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.790.3.6** virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.790.3.7** virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SubscriptionInfoMarshaller.h**

## 6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3692).

#include <src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.791.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3692).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.791.2 Constructor & Destructor Documentation

**6.791.2.1** `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

**6.791.2.2** `virtual activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

## 6.791.3 Member Function Documentation

**6.791.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.791.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.791.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.791.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.791.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.791.3.6** virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

**6.791.3.7** virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**SubscriptionInfoMarshaller.h**

## 6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3696).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.792.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3696).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.792.2 Constructor & Destructor Documentation

**6.792.2.1** `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

**6.792.2.2** `virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

## 6.792.3 Member Function Documentation

**6.792.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.792.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.792.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.792.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.792.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1** (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.792.3.6** virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.792.3.7** virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

## 6.793 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

#include <src/main/decaf/util/concurrent/Synchronizable.h> Inheritance diagram for decaf::util::concurrent::Synchronizable:

### Public Member Functions

- virtual **~Synchronizable** ()
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to **Lock** (p. 2375) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

## 6.793.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since:

1.0

## 6.793.2 Constructor & Destructor Documentation

**6.793.2.1** virtual decaf::util::concurrent::Synchronizable::~~Synchronizable ()  
[inline, virtual]

## 6.793.3 Member Function Documentation

**6.793.3.1** virtual void decaf::util::concurrent::Synchronizable::lock () throw (  
decaf::lang::exceptions::RuntimeException ) [pure virtual]

Locks the object.

Exceptions:

*RuntimeException* if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2602), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3712), `decaf::io::InputStream` (p. 2046), `decaf::io::OutputStream` (p. 2910), `decaf::util::AbstractCollection< E >` (p. 186), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1241), `decaf::util::concurrent::Mutex` (p. 2783), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3609), `decaf::util::StlQueue< T >` (p. 3619), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 186), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 186), `decaf::util::AbstractCollection< Resource * >` (p. 186), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 186), `decaf::util::AbstractCollection< CompositeTask * >` (p. 186), `decaf::util::AbstractCollection< URI >` (p. 186), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 186), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 186), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 186), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 186), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 186), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 186), `decaf::util::AbstractCollection< cms::Destination * >` (p. 186), `decaf::util::AbstractCollection< cms::Session * >` (p. 186), `decaf::util::AbstractCollection< cms::Connection * >` (p. 186), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1241), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1241), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1241), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1241), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1241),

decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1241), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3609), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3609), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3609), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3609), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3609), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< int, Pointer< Command > > (p. 3609), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3609), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3609), decaf::util::StlMap< std::string, cms::Topic \* > (p. 3609), decaf::util::StlQueue< Pointer< Transport > > (p. 3619), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3619), decaf::util::StlQueue< Task > (p. 3619), decaf::util::StlQueue< Pointer< Command > > (p. 3619), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3619).

**6.793.3.2** virtual void decaf::util::concurrent::Synchronizable::notify  
 () throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

***IllegalMonitorStateException*** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

***RuntimeException*** if an error occurs while notifying one of the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2602), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3712), **decaf::io::InputStream** (p. 2047), **decaf::io::OutputStream** (p. 2910), **decaf::util::AbstractCollection< E >** (p. 186), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1241), **decaf::util::concurrent::Mutex** (p. 2783), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3610), **decaf::util::StlQueue< T >** (p. 3619), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 186), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 186), **decaf::util::AbstractCollection< Resource \* >** (p. 186), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 186), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 186), **decaf::util::AbstractCollection< URI >** (p. 186), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 186), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 186), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 186), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 186), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 186), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 186), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 186), **decaf::util::AbstractCollection< cms::Session**

\* > (p. 186), decaf::util::AbstractCollection< cms::Connection \* > (p. 186), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1241), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3610), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3610), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3610), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3610), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3610), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3610), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< int, Pointer< Command > > (p. 3610), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3610), decaf::util::StlMap< std::string, cms::Topic \* > (p. 3610), decaf::util::StlQueue< Pointer< Transport > > (p. 3619), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3619), decaf::util::StlQueue< Task > (p. 3619), decaf::util::StlQueue< Pointer< Command > > (p. 3619), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3619).

**6.793.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll**  
 () throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

#### Exceptions:

**IllegalMonitorStateException** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

**RuntimeException** if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2602), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3712), **decaf::io::InputStream** (p. 2047), **decaf::io::OutputStream** (p. 2911), **decaf::util::AbstractCollection< E >** (p. 187), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1241), **decaf::util::concurrent::Mutex** (p. 2784), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3610), **decaf::util::StlQueue< T >** (p. 3619), **decaf::util::AbstractCollection< transport::TransportListener**

\* > (p. 187), decaf::util::AbstractCollection< Pointer< Synchroniza-  
 tion > > (p. 187), decaf::util::AbstractCollection< Resource \* >  
 (p. 187), decaf::util::AbstractCollection< cms::MessageConsumer \* >  
 (p. 187), decaf::util::AbstractCollection< CompositeTask \* > (p. 187),  
 decaf::util::AbstractCollection< URI > (p. 187), decaf::util::AbstractCollection<  
 ActiveMQSession \* > (p. 187), decaf::util::AbstractCollection< Pointer<  
 DestinationInfo > > (p. 187), decaf::util::AbstractCollection< Prim-  
 itiveValueNode > > (p. 187), decaf::util::AbstractCollection< Pointer<  
 Command > > (p. 187), decaf::util::AbstractCollection< Pointer<  
 BackupTransport > > (p. 187), decaf::util::AbstractCollection<  
 cms::MessageProducer \* > (p. 187), decaf::util::AbstractCollection<  
 cms::Destination \* > (p. 187), decaf::util::AbstractCollection< cms::Session  
 \* > (p. 187), decaf::util::AbstractCollection< cms::Connection \* >  
 (p. 187), decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes-  
 sageId >, Pointer< Message >, MessageId::COMPARATOR >  
 (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-  
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR  
 > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-  
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR  
 > (p. 1241), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId  
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1241),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,  
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1241),  
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<  
 ProducerState >, ProducerId::COMPARATOR > (p. 1241), decaf::util::StlMap<  
 cms::Session \*, SessionResolver \* > (p. 3610), decaf::util::StlMap< std::string,  
 WireFormatFactory \* > (p. 3610), decaf::util::StlMap< std::string, PrimitiveVal-  
 ueNode > (p. 3610), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3610),  
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, com-  
 mands::ProducerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string,  
 CachedConsumer \* > (p. 3610), decaf::util::StlMap< Pointer< commands::ConsumerId  
 >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3610),  
 decaf::util::StlMap< std::string, TransportFactory \* > (p. 3610), decaf::util::StlMap<  
 Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR  
 > (p. 3610), decaf::util::StlMap< int, Pointer< Command > > (p. 3610),  
 decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, com-  
 mands::ConsumerId::COMPARATOR > (p. 3610), decaf::util::StlMap< std::string,  
 CachedProducer \* > (p. 3610), decaf::util::StlMap< std::string, cms::Topic  
 \* > (p. 3610), decaf::util::StlQueue< Pointer< Transport > > (p. 3619),  
 decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3619), decaf::util::StlQueue<  
 Task > (p. 3619), decaf::util::StlQueue< Pointer< Command > > (p. 3619), and  
 decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >  
 (p. 3619).

#### 6.793.3.4 virtual bool decaf::util::concurrent::Synchronizable::tryLock () throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Attempts to **Lock** (p. 2375) the object, if the lock is already held by another thread than this method returns false.

#### Returns:

true if the lock was acquired, false if it is already held by another thread.



**Exceptions:**

*RuntimeException* if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2603), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3713), `decaf::io::InputStream` (p. 2051), `decaf::io::OutputStream` (p. 2911), `decaf::util::AbstractCollection< E >` (p. 189), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1246), `decaf::util::concurrent::Mutex` (p. 2784), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3612), `decaf::util::StlQueue< T >` (p. 3620), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 189), `decaf::util::AbstractCollection< Pointer< Synchroniza- tion > >` (p. 189), `decaf::util::AbstractCollection< Resource * >` (p. 189), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 189), `decaf::util::AbstractCollection< CompositeTask * >` (p. 189), `decaf::util::AbstractCollection< URI >` (p. 189), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 189), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 189), `decaf::util::AbstractCollection< Prim- itiveValueNode >` (p. 189), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 189), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 189), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 189), `decaf::util::AbstractCollection< cms::Destination * >` (p. 189), `decaf::util::AbstractCollection< cms::Session * >` (p. 189), `decaf::util::AbstractCollection< cms::Connection * >` (p. 189), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes- sageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec- tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con- sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1246), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3612), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3612), `decaf::util::StlMap< std::string, PrimitiveVal- ueNode >` (p. 3612), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com- mands::ProducerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3612), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, com- mands::ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3612), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3612), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3620), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3620), `decaf::util::StlQueue< Task >` (p. 3620), `decaf::util::StlQueue< Pointer< Command > >` (p. 3620), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3620).

**6.793.3.5** virtual void decaf::util::concurrent::Synchronizable::unlock () throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Unlocks the object.

#### Exceptions:

*RuntimeException* if an error occurs while unlocking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2604), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3713), `decaf::io::InputStream` (p. 2051), `decaf::io::OutputStream` (p. 2911), `decaf::util::AbstractCollection< E >` (p. 190), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1246), `decaf::util::concurrent::Mutex` (p. 2784), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3612), `decaf::util::StlQueue< T >` (p. 3621), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 190), `decaf::util::AbstractCollection< Pointer< Synchroniza- tion > >` (p. 190), `decaf::util::AbstractCollection< Resource * >` (p. 190), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 190), `decaf::util::AbstractCollection< CompositeTask * >` (p. 190), `decaf::util::AbstractCollection< URI >` (p. 190), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 190), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 190), `decaf::util::AbstractCollection< Prim- itiveValueNode >` (p. 190), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 190), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 190), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 190), `decaf::util::AbstractCollection< cms::Destination * >` (p. 190), `decaf::util::AbstractCollection< cms::Session * >` (p. 190), `decaf::util::AbstractCollection< cms::Connection * >` (p. 190), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes- sageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec- tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con- sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1246), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1246), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3612), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3612), `decaf::util::StlMap< std::string, PrimitiveVal- ueNode >` (p. 3612), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com- mands::ProducerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3612), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3612), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, com- mands::ConsumerId::COMPARATOR >` (p. 3612), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3612), `decaf::util::StlMap< std::string, cms::Topic`

\* > (p. 3612), decaf::util::StlQueue< Pointer< Transport > > (p. 3621), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3621), decaf::util::StlQueue< Task > (p. 3621), decaf::util::StlQueue< Pointer< Command > > (p. 3621), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3621).

**6.793.3.6** virtual void decaf::util::concurrent::Synchronizable::wait (long long *milliseconds*, int *nanos*) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters:

*milliseconds* the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

#### Exceptions:

**IllegalArgumentException** if an error occurs or the nanos argument is not in the range of [0-999999]

**RuntimeException** if an error occurs while waiting on the object.

**InterruptedException** if the wait is interrupted before it completes.

**IllegalMonitorStateException** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2604), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3713), **decaf::io::InputStream** (p. 2051), **decaf::io::OutputStream** (p. 2912), **decaf::util::AbstractCollection< E >** (p. 190), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1247), **decaf::util::concurrent::Mutex** (p. 2785), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3613), **decaf::util::StlQueue< T >** (p. 3621), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 190), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 190), **decaf::util::AbstractCollection< Resource \* >** (p. 190), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 190), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 190), **decaf::util::AbstractCollection< URI >** (p. 190), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 190), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 190), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 190), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 190), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 190), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 190), **decaf::util::AbstractCollection<**

`cms::Destination * > (p. 190), decaf::util::AbstractCollection< cms::Session * > (p. 190), decaf::util::AbstractCollection< cms::Connection * > (p. 190), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1247), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3613), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3613), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3613), decaf::util::StlMap< std::string, cms::Queue * > (p. 3613), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3613), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3613), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3613), decaf::util::StlMap< std::string, TransportFactory * > (p. 3613), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3613), decaf::util::StlMap< int, Pointer< Command > > (p. 3613), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3613), decaf::util::StlMap< std::string, CachedProducer * > (p. 3613), decaf::util::StlMap< std::string, cms::Topic * > (p. 3613), decaf::util::StlQueue< Pointer< Transport > > (p. 3621), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3621), decaf::util::StlQueue< Task > (p. 3621), decaf::util::StlQueue< Pointer< Command > > (p. 3621), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3621).`

**6.793.3.7** `virtual void decaf::util::concurrent::Synchronizable::wait (long long milliseconds) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters:

*milliseconds* the time in milliseconds to wait, or WAIT\_INFINITE

#### Exceptions:

**RuntimeException** if an error occurs while waiting on the object.

**InterruptedException** if the wait is interrupted before it completes.

**IllegalMonitorStateException** - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2604), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3714), `decaf::io::InputStream`

(p. 2052), decaf::io::OutputStream (p. 2912), decaf::util::AbstractCollection< E > (p. 191), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1247), decaf::util::concurrent::Mutex (p. 2785), decaf::util::StlMap< K, V, COMPARATOR > (p. 3613), decaf::util::StlQueue< T > (p. 3621), decaf::util::AbstractCollection< transport::TransportListener \* > (p. 191), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 191), decaf::util::AbstractCollection< Resource \* > (p. 191), decaf::util::AbstractCollection< cms::MessageConsumer \* > (p. 191), decaf::util::AbstractCollection< CompositeTask \* > (p. 191), decaf::util::AbstractCollection< URI > (p. 191), decaf::util::AbstractCollection< ActiveMQSession \* > (p. 191), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 191), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 191), decaf::util::AbstractCollection< Pointer< Command > > (p. 191), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 191), decaf::util::AbstractCollection< cms::MessageProducer \* > > (p. 191), decaf::util::AbstractCollection< cms::Destination \* > > (p. 191), decaf::util::AbstractCollection< cms::Session \* > > (p. 191), decaf::util::AbstractCollection< cms::Connection \* > > (p. 191), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > > (p. 1247), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > > (p. 1247), decaf::util::StlMap< cms::Session \*, SessionResolver \* > > (p. 3613), decaf::util::StlMap< std::string, WireFormatFactory \* > > (p. 3613), decaf::util::StlMap< std::string, PrimitiveValueNode > > (p. 3613), decaf::util::StlMap< std::string, cms::Queue \* > > (p. 3613), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > > (p. 3613), decaf::util::StlMap< std::string, CachedConsumer \* > > (p. 3613), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > > (p. 3613), decaf::util::StlMap< std::string, TransportFactory \* > > (p. 3613), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > > (p. 3613), decaf::util::StlMap< int, Pointer< Command > > > (p. 3613), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > > (p. 3613), decaf::util::StlMap< std::string, CachedProducer \* > > (p. 3613), decaf::util::StlMap< std::string, cms::Topic \* > > (p. 3613), decaf::util::StlQueue< Pointer< Transport > > > (p. 3621), decaf::util::StlQueue< Pointer< MessageDispatch > > > (p. 3621), decaf::util::StlQueue< Task > > (p. 3621), decaf::util::StlQueue< Pointer< Command > > > (p. 3621), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > > (p. 3621).

**6.793.3.8** virtual void decaf::util::concurrent::Synchronizable::wait  
 () throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException,  
 decaf::lang::exceptions::InterruptedException ) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

#### Exceptions:

*RuntimeException* if an error occurs while waiting on the object.

*InterruptedException* if the wait is interrupted before it completes.

*IllegalMonitorStateException* - if the current thread is not the owner of the the **Synchronizable** (p. 3700) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2605), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3714), **decaf::io::InputStream** (p. 2052), **decaf::io::OutputStream** (p. 2913), **decaf::util::AbstractCollection< E >** (p. 191), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1248), **decaf::util::concurrent::Mutex** (p. 2786), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3614), **decaf::util::StlQueue< T >** (p. 3622), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 191), **decaf::util::AbstractCollection< Pointer< Synchroniza- tion > >** (p. 191), **decaf::util::AbstractCollection< Resource \* >** (p. 191), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 191), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 191), **decaf::util::AbstractCollection< URI >** (p. 191), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 191), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 191), **decaf::util::AbstractCollection< Prim- itiveValueNode >** (p. 191), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 191), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 191), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 191), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 191), **decaf::util::AbstractCollection< cms::Session \* >** (p. 191), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 191), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes- sageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1248), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec- tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1248), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Con- sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1248), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1248), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1248), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1248), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3614), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3614), **decaf::util::StlMap< std::string, PrimitiveVal- ueNode >** (p. 3614), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3614), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, com- mands::ProducerId::COMPARATOR >** (p. 3614), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3614), **decaf::util::StlMap< Pointer< commands::ConsumerId**

>, `ActiveMQConsumer *`, `commands::ConsumerId::COMPARATOR` > (p. 3614), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3614), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3614), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3614), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3614), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3614), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3614), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3622), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3622), `decaf::util::StlQueue< Task >` (p. 3622), `decaf::util::StlQueue< Pointer< Command > >` (p. 3622), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3622).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Synchronizable.h`

## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h> Inheritance diagram for decaf::internal::util::concurrent::SynchronizableImpl:

### Public Member Functions

- **SynchronizableImpl** ()
- virtual **~SynchronizableImpl** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*



### 6.794.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since:

1.0

### 6.794.2 Constructor & Destructor Documentation

**6.794.2.1** `decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl()`

**6.794.2.2** `virtual decaf::internal::util::concurrent::SynchronizableImpl::~SynchronizableImpl() [virtual]`

### 6.794.3 Member Function Documentation

**6.794.3.1** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock () throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Locks the object.

Exceptions:

*RuntimeException* if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3701).

**6.794.3.2** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

*IllegalMonitorStateException* - if the current thread is not the owner of the the Synchronizable Object.

*RuntimeException* if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3702).

**6.794.3.3** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

**Exceptions:**

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

***RuntimeException*** if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3703).

**6.794.3.4** `virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock()  
( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns:**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions:**

***RuntimeException*** if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3704).

**6.794.3.5** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock()  
( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Unlocks the object.

**Exceptions:**

***RuntimeException*** if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3705).

**6.794.3.6** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait  
(long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters:**

***millisecs*** the time in milliseconds to wait, or WAIT\_INFINITE

*nanos* additional time in nanoseconds with a range of 0-999999

#### Exceptions:

***IllegalArgumentException*** if an error occurs or the nanos argument is not in the range of [0-999999]

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3707).

**6.794.3.7** virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long *millisecs*) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters:

*millisecs* the time in milliseconds to wait, or WAIT\_INFINITE

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3708).

**6.794.3.8** virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

#### Exceptions:

***RuntimeException*** if an error occurs while waiting on the object.

***InterruptedException*** if the wait is interrupted before it completes.

***IllegalMonitorStateException*** - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3709).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**SynchronizableImpl.h**

## 6.795 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 3715), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

### Public Member Functions

- virtual **~Synchronization** ()
- virtual void **beforeEnd** ()=0 throw ( exceptions::ActiveMQException )
- virtual void **afterCommit** ()=0 throw ( exceptions::ActiveMQException )
- virtual void **afterRollback** ()=0 throw ( exceptions::ActiveMQException )

### 6.795.1 Detailed Description

Transacted Object **Synchronization** (p. 3715), used to sync the events of a Transaction with the items in the Transaction.

### 6.795.2 Constructor & Destructor Documentation

**6.795.2.1** virtual **activemq::core::Synchronization::~~Synchronization** () [inline, virtual]

### 6.795.3 Member Function Documentation

**6.795.3.1** virtual void **activemq::core::Synchronization::afterCommit** () throw ( exceptions::ActiveMQException ) [pure virtual]

**6.795.3.2** virtual void **activemq::core::Synchronization::afterRollback** () throw ( exceptions::ActiveMQException ) [pure virtual]

**6.795.3.3** virtual void **activemq::core::Synchronization::beforeEnd** () throw ( exceptions::ActiveMQException ) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Synchronization.h**

## 6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A **blocking queue** (p. 843) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

#include <src/main/decaf/util/concurrent/SynchronousQueue.h> Inheritance diagram for decaf::util::concurrent::SynchronousQueue< E >:

### Data Structures

- class **EmptyIterator**

### Public Member Functions

- **SynchronousQueue** ()
- virtual **~SynchronousQueue** ()
- virtual void **put** (const E &value) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Adds the specified element to this queue, waiting if necessary for another thread to receive it.*
- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, if another thread is waiting to receive it.*
- virtual E **take** () throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.*
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.*
- virtual bool **poll** (E &result)  
*Retrieves and removes the head of this queue, if another thread is currently making an element available.*
- virtual bool **equals** (const **Collection**< E > &value) const  
*Answers true if this **Collection** (p. 1184) and the one given are the same size and if each element contained in the **Collection** (p. 1184) given is equal to an element contained in this collection.*

- virtual **decaf::util::Iterator**< E > \* **iterator** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual bool **isEmpty** () const  
*Returns true if this collection contains no elements.*
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual int **remainingCapacity** () const  
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX\_VALUE** if there is no intrinsic limit.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **contains** (const E &value DECAF\_UNUSED) const throw ( lang::Exception )
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **remove** (const E &value DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **removeAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **retainAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **peek** (E &result DECAF\_UNUSED) const
- virtual std::vector< E > **toArray** () const  
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1184).*
- virtual std::size\_t **drainTo** (**Collection**< E > &c) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes all available elements from this queue and adds them to the given collection.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes at most the given number of available elements from this queue and adds them to the given collection.*

## 6.796.1 Detailed Description

**template<typename E> class decaf::util::concurrent::SynchronousQueue< E >**

A **blocking queue** (p. 843) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any **internal**

(p. 144) capacity, not even a capacity of one. You cannot **peek** at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and **poll()** (p. 3722) will return **null**. For purposes of other **Collection** (p. 1184) methods (for example **contains**), a **SynchronousQueue** (p. 3716) acts as an empty collection. This queue does not permit **null** elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to **true** grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p. 1184) and **Iterator** (p. 2154) interfaces.

Since:

1.0

## 6.796.2 Constructor & Destructor Documentation

**6.796.2.1** `template<typename E> decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue () [inline]`

**6.796.2.2** `template<typename E> virtual decaf::util::concurrent::SynchronousQueue< E >::~~SynchronousQueue () [inline, virtual]`

## 6.796.3 Member Function Documentation

**6.796.3.1** `template<typename E> virtual void decaf::util::concurrent::SynchronousQueue< E >::clear () throw ( lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all elements of the queue. This implementation repeatedly invokes **poll** until it returns the empty marker.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 196).

**6.796.3.2** `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::contains  
(const E &value DECAF_UNUSED) const throw ( lang::Exception )  
[inline, virtual]`

**6.796.3.3** `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::containsAll (const Collection< E > & collection) const throw (   
lang::Exception ) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection. This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

**Parameters:**

*collection* collection to be checked for containment in this collection

**Returns:**

true if this collection contains all of the elements in the specified collection.

**Exceptions:**

*Exception* if an error occurs,

Reimplemented from `decaf::util::AbstractCollection< E >` (p.185).

**6.796.3.4** `template<typename E > virtual std::size_t  
decaf::util::concurrent::SynchronousQueue< E >::drainTo  
(Collection< E > & c, std::size_t maxElements) throw  
( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes at most the given number of available elements from this queue and adds them to the given collection. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Parameters:**

*c* the collection to transfer elements into

*maxElements* the maximum number of elements to transfer

**Returns:**

the number of elements transferred

**Exceptions:**

*UnsupportedOperationException* if addition of elements is not supported by the specified collection

*IllegalArgumentException* if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection



## 6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference 8723

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 846).

References **decaf::util::AbstractQueue< E >::element()**, and **decaf::util::concurrent::SynchronousQueue< E >::poll()**.

**6.796.3.5** `template<typename E> virtual std::size_t  
decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E  
> & c) throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes all available elements from this queue and adds them to the given collection. This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in **IllegalArgumentException**. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

### Parameters:

*c* the collection to transfer elements into

### Returns:

the number of elements transferred

### Exceptions:

**UnsupportedOperationException** if addition of elements is not supported by the specified collection

**IllegalArgumentException** if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 846).

References **decaf::util::AbstractQueue< E >::element()**, and **decaf::util::concurrent::SynchronousQueue< E >::poll()**.

**6.796.3.6** `template<typename E> virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::equals  
(const Collection< E > & collection) const [inline, virtual]`

Answers true if this **Collection** (p. 1184) and the one given are the same size and if each element contained in the **Collection** (p. 1184) given is equal to an element contained in this collection.

### Parameters:

*collection* - The **Collection** (p. 1184) to be compared to this one.

### Returns:

true if this **Collection** (p. 1184) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 185).

**6.796.3.7** `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::isEmpty  
( ) const [inline, virtual]`

Returns true if this collection contains no elements. This implementation returns `size()` (p. 3724) == 0.

**Returns:**

true if the size method return 0.

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 186).

**6.796.3.8** `template<typename E > virtual decaf::util::Iterator<E>*  
decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) const  
[inline, virtual]`

Implements `decaf::lang::Iterable< E >` (p. 2152).

**6.796.3.9** `template<typename E > virtual decaf::util::Iterator<E>*  
decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) [inline,  
virtual]`

**Returns:**

an iterator over a set of elements of type T.

Implements `decaf::lang::Iterable< E >` (p. 2152).

**6.796.3.10** `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::offer  
(const E & value) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Inserts the specified element into this queue, if another thread is waiting to receive it.

**Parameters:**

*value* the element to add to the **Queue** (p. 3149)

**Returns:**

true if the element was added to this queue, else false

**Exceptions:**

*NullPointerException* if the **Queue** (p. 3149) implementation does not allow Null values to be inserted into the **Queue** (p. 3149).

*IllegalArgumentException* if some property of the specified element prevents it from being added to this queue

Implements `decaf::util::Queue< E >` (p. 3150).

**6.796.3.11**    `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::offer  
(const E & e, long timeout, const TimeUnit & unit)  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException, de-  
caf::lang::exceptions::IllegalArgumentException ) [inline,  
virtual]`

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

**Returns:**

`true` if successful, or `false` if the specified waiting time elapses before a consumer appears.

**Exceptions:**

***InterruptedException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

***NullPointerException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

***IllegalArgumentException*** Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 847).

**6.796.3.12**    `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::peek (E  
&result DECAF_UNUSED) const [inline, virtual]`

**6.796.3.13**    `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::poll (E  
& result) [inline, virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

**Parameters:**

*result* a reference to the value where the head of the **Queue** (p. 3149) should be copied to.

**Returns:**

`true` if the head of the **Queue** (p. 3149) was copied to the result param or `false` if no value could be returned.

Implements `decaf::util::Queue< E >` (p. 3151).

**6.796.3.14**    `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::poll (E  
& result, long long timeout, const TimeUnit & unit) throw (  
decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

**Parameters:**

**result** a reference to the value where the head of the **Queue** (p. 3149) should be copied to.

**timeout** the time that the method should block if there is no element available to return.

**unit** the Time Units that the timeout value represents.

**Returns:**

true if the head of the **Queue** (p. 3149) was copied to the result param or false if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 847).

Referenced by **decaf::util::concurrent::SynchronousQueue< E >::drainTo()**.

```
6.796.3.15  template<typename E > virtual void
             decaf::util::concurrent::SynchronousQueue< E >::put
             (const E & value) throw ( decaf::lang::exceptions::InterruptedException,
             decaf::lang::exceptions::NullPointerException, de-
             ccaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

**Parameters:**

**value** the element to add to the **Queue** (p. 3149).

**Exceptions:**

**InterruptedException** Inserts the specified element into this queue, waiting if necessary for space to become available.

**NullPointerException** Inserts the specified element into this queue, waiting if necessary for space to become available.

**IllegalArgumentException** Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 848).

```
6.796.3.16  template<typename E > virtual int
             decaf::util::concurrent::SynchronousQueue< E
             >::remainingCapacity () const [inline, virtual]
```

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or **Integer::MAX\_VALUE** if there is no intrinsic limit. Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting **remainingCapacity** because it may be the case that another thread is about to insert or remove an element.

**Returns:**

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 848).

- 6.796.3.17** `template<typename E> virtual bool  
decaf::util::concurrent::SynchronousQueue< E >::remove  
(const E &value DECAF_UNUSED) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline, virtual]`
- 6.796.3.18** `template<typename E> virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::removeAll (const Collection< E > &collection DECAF_UNUSED)  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline, virtual]`
- 6.796.3.19** `template<typename E> virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::retainAll (const Collection< E > &collection DECAF_UNUSED)  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline, virtual]`
- 6.796.3.20** `template<typename E> virtual std::size_t  
decaf::util::concurrent::SynchronousQueue< E >::size () const [inline,  
virtual]`

Returns the number of elements in this collection. If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**Returns:**

the number of elements in this collection

Implements `decaf::util::Collection< E >` (p. 1192).

- 6.796.3.21** `template<typename E> virtual E  
decaf::util::concurrent::SynchronousQueue< E >::take ()  
throw ( decaf::lang::exceptions::InterruptedException ) [inline,  
virtual]`

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

**Returns:**

the head of this queue

**Exceptions:**

*InterruptedException* Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 849).

- 6.796.3.22** `template<typename E> virtual std::vector<E>  
decaf::util::concurrent::SynchronousQueue< E >::toArray () const  
[inline, virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1184). All the elements in the array will not be referenced by the collection. The elements in the returned

array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

**Returns:**

an vector of copies of all the elements from this **Collection** (p. 1184)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 189).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/SynchronousQueue.h`

## 6.797 decaf::lang::System Class Reference

The **System** (p.3726) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

```
#include <src/main/decaf/lang/System.h>
```

### Public Member Functions

- virtual `~System()`

### Static Public Member Functions

- static void **arraycopy** (const unsigned char \*src, std::size\_t srcPos, unsigned char \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static void **arraycopy** (const short \*src, std::size\_t srcPos, short \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static void **arraycopy** (const int \*src, std::size\_t srcPos, int \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static void **arraycopy** (const long long \*src, std::size\_t srcPos, long long \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static const **util::Map**< std::string, std::string > & **getenv** ()  
*Enumerates the system environment and returns a map of env variable names to the string values they hold.*
- static std::string **getenv** (const std::string &name)  
*Reads an environment value from the system and returns it as a string object.*
- static void **unsetenv** (const std::string &name)  
*Clears a set environment value if one is set.*
- static void **setenv** (const std::string &name, const std::string &value)  
*Sets the specified system property to the value given.*
- static long long **currentTimeMillis** ()

*Returns the current time in milliseconds.*

- static long long **nanoTime** ()

*Returns the current value of the most precise available system timer, in nanoseconds.*

- static int **availableProcessors** ()

*Returns the number of processors available for execution of Decaf Threads.*

- static **decaf::util::Properties** & **getProperties** ()

*Gets the Properties object that holds the Properties accessed from calls to **getProperty** and **setProperty**.*

- static std::string **getProperty** (const std::string &key)

*Gets the specified **System** (p. 3726) property if set, otherwise returns an empty string.*

- static std::string **getProperty** (const std::string &key, const std::string &defaultValue)

*Gets the specified **System** (p. 3726) property if set, otherwise returns the specified default value.*

- static std::string **setProperty** (const std::string &key, const std::string &value)

*Sets the **System** (p. 3726) Property to the specified value.*

- static std::string **clearProperty** (const std::string &key)

*Clear any value associated with the system property specified.*

## Protected Member Functions

- **System** ()

## Friends

- class **decaf::lang::Runtime**

### 6.797.1 Detailed Description

The **System** (p. 3726) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

**Since:**

1.0



## 6.797.2 Constructor & Destructor Documentation

**6.797.2.1** `decaf::lang::System::System ()` [protected]

**6.797.2.2** `virtual decaf::lang::System::~~System ()` [inline, virtual]

## 6.797.3 Member Function Documentation

**6.797.3.1** `static void decaf::lang::System::arraycopy (const long long * src, std::size_t srcPos, long long * dest, std::size_t destPos, std::size_t length)` [static]

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

### Parameters:

*src* The source array to copy from.

*srcPos* The position in the array to start copying from.

*dest* The destination array to copy to.

*destPos* The position in the destination array to start writing at.

*length* The number of elements to copy from *src* to *dest*.

### Exceptions:

*NullPointerException* if *src* or *dest* are NULL.

**6.797.3.2** `static void decaf::lang::System::arraycopy (const int * src, std::size_t srcPos, int * dest, std::size_t destPos, std::size_t length)` [static]

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

### Parameters:

*src* The source array to copy from.

*srcPos* The position in the array to start copying from.

*dest* The destination array to copy to.

*destPos* The position in the destination array to start writing at.

*length* The number of elements to copy from *src* to *dest*.

### Exceptions:

*NullPointerException* if *src* or *dest* are NULL.

**6.797.3.3** `static void decaf::lang::System::arraycopy (const short * src, std::size_t srcPos, short * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

**Parameters:**

*src* The source array to copy from.  
*srcPos* The position in the array to start copying from.  
*dest* The destination array to copy to.  
*destPos* The position in the destination array to start writing at.  
*length* The number of elements to copy from *src* to *dest*.

**Exceptions:**

*NullPointerException* if *src* or *dest* are NULL.

**6.797.3.4** `static void decaf::lang::System::arraycopy (const unsigned char * src, std::size_t srcPos, unsigned char * dest, std::size_t destPos, std::size_t length) [static]`

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

**Parameters:**

*src* The source array to copy from.  
*srcPos* The position in the array to start copying from.  
*dest* The destination array to copy to.  
*destPos* The position in the destination array to start writing at.  
*length* The number of elements to copy from *src* to *dest*.

**Exceptions:**

*NullPointerException* if *src* or *dest* are NULL.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`.

**6.797.3.5** `static int decaf::lang::System::availableProcessors () [static]`

Returns the number of processors available for execution of Decaf Threads. This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

**Returns:**

the number of available processors.

**6.797.3.6** static std::string decaf::lang::System::clearProperty (const std::string & *key*) [static]

Clear any value associated with the system property specified.

**Parameters:**

*key* The key name of the system property to clear.

**Returns:**

the previous value of the property named by key if there was one, otherwise returns an empty string.

**Exceptions:**

*IllegalArgumentException* if key is an empty string.

**6.797.3.7** static long long decaf::lang::System::currentTimeMillis () [static]

Returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

**Returns:**

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

**6.797.3.8** static std::string decaf::lang::System::getenv (const std::string & *name*) [static]

Reads an environment value from the system and returns it as a string object.

**Parameters:**

*name* The environment variable to read.

**Returns:**

a string with the value from the variables or ""

**Exceptions:**

*an Exception* (p. 1831) if an error occurs while reading the Env.

**6.797.3.9** static const util::Map<std::string, std::string>& decaf::lang::System::getenv () [static]

Enumerates the system environment and returns a map of env variable names to the string values they hold.

**Returns:**

A Map of all environment variables.

**Exceptions:**

*Exception* (p. 1831) if an error occurs while getting the Environment Map.

**6.797.3.10** `static decaf::util::Properties& decaf::lang::System::getProperties ()`  
[static]

Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty. If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

**Returns:**

a reference to the static system Properties object.

**6.797.3.11** `static std::string decaf::lang::System::getProperty (const std::string &key, const std::string & defaultValue)` [static]

Gets the specified **System** (p. 3726) property if set, otherwise returns the specified default value. If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

**Parameters:**

*key* The key name of the desired system property to retrieve.

*defaultValue* The default value to return if the key is not set in the **System** (p. 3726) properties.

**Returns:**

the value of the named system property or the defaultValue if the property isn't set..

**Exceptions:**

*IllegalArgumentException* if key is an empty string.

**6.797.3.12** `static std::string decaf::lang::System::getProperty (const std::string &key)` [static]

Gets the specified **System** (p. 3726) property if set, otherwise returns an empty string. If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

**Parameters:**

*key* The key name of the desired system property to retrieve.

**Returns:**

an empty string if the named property is not set, otherwise returns the value.

**Exceptions:**

*IllegalArgumentException* if key is an empty string.

**6.797.3.13 static long long decaf::lang::System::nanoTime () [static]**

Returns the current value of the most precise available system timer, in nanoseconds. This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some **code** (p. 1183) takes to execute:

```
long long startTime = System::nanoTime() (p. 3732); // ... the code (p. 1183) being measured
... long long estimatedTime = System::nanoTime() (p. 3732) - startTime;
```

**Returns:**

The current value of the system timer, in nanoseconds.

**6.797.3.14 static void decaf::lang::System::setenv (const std::string & name, const std::string & value) [static]**

Sets the specified system property to the value given.

**Parameters:**

*name* The name of the environment variables to set.

*value* The value to assign to name.

**Exceptions:**

*an Exception* (p. 1831) if an error occurs when setting the environment variable.

**6.797.3.15 static std::string decaf::lang::System::setProperty (const std::string & key, const std::string & value) [static]**

Sets the **System** (p. 3726) Property to the specified value.

**Parameters:**

*key* The key name of the system property to set to the given value.

*value* The value to assign to the key.

**Returns:**

the previous value of the property named by key if there was one, otherwise returns an empty string.

**Exceptions:**

*IllegalArgumentException* if key is an empty string.

**6.797.3.16**   `static void decaf::lang::System::unsetenv (const std::string & name)`  
                  `[static]`

Clears a set environment value if one is set.

**Parameters:**

*name*   The environment variables to clear.

**Exceptions:**

*an Exception* (p. 1831) if an error occurs while reading the environment.

## 6.797.4   Friends And Related Function Documentation

**6.797.4.1**   `friend class decaf::lang::Runtime`   `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/System.h`

## 6.798 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

#include <src/main/activemq/threads/Task.h> Inheritance diagram for activemq::threads::Task:

### Public Member Functions

- virtual `~Task()`
- virtual `bool iterate()` = 0

*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.798.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since:

3.0

### 6.798.2 Constructor & Destructor Documentation

**6.798.2.1** virtual `activemq::threads::Task::~~Task()` [inline, virtual]

### 6.798.3 Member Function Documentation

**6.798.3.1** virtual `bool activemq::threads::Task::iterate()` [pure virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 535), `activemq::threads::CompositeTaskRunner` (p. 1224), `activemq::transport::failover::BackupTransportPool` (p. 757), `activemq::transport::failover::CloseTransportsTask` (p. 1152), and `activemq::transport::failover::FailoverTransport` (p. 1879).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

## 6.799 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

### Public Member Functions

- virtual `~TaskListener ()`
- virtual void `onTaskComplete (lang::Runnable *task)=0`  
*Called when a queued task has completed, the task that finished is passed along for user consumption.*
- virtual void `onTaskException (lang::Runnable *task, lang::Exception &ex)=0`  
*Called when a queued task has thrown an exception while being run.*

### 6.799.1 Constructor & Destructor Documentation

- 6.799.1.1** virtual `decaf::util::concurrent::TaskListener::~~TaskListener ()` [inline, virtual]

### 6.799.2 Member Function Documentation

- 6.799.2.1** virtual void `decaf::util::concurrent::TaskListener::onTaskComplete (lang::Runnable * task)` [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

#### Parameters:

*task* Runnable Pointer to the task that finished

- 6.799.2.2** virtual void `decaf::util::concurrent::TaskListener::onTaskException (lang::Runnable * task, lang::Exception & ex)` [pure virtual]

Called when a queued task has thrown an exception while being run. The Callee should assume that this was an unrecoverable exception and that this task is now defunct.

#### Parameters:

*task* Runnable Pointer to the task

*ex* The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TaskListener.h`



## 6.800 activemq::threads::TaskRunner Class Reference

#include <src/main/activemq/threads/TaskRunner.h> Inheritance diagram for activemq::threads::TaskRunner:

### Public Member Functions

- virtual `~TaskRunner ()`
- virtual void **shutdown** (unsigned int timeout)=0  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()=0  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()=0  
*Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.*

### 6.800.1 Constructor & Destructor Documentation

**6.800.1.1** virtual activemq::threads::TaskRunner::~~TaskRunner () [inline, virtual]

### 6.800.2 Member Function Documentation

**6.800.2.1** virtual void activemq::threads::TaskRunner::shutdown () [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1224), and `activemq::threads::DedicatedTaskRunner` (p. 1672).

**6.800.2.2** virtual void activemq::threads::TaskRunner::shutdown (unsigned int *timeout*) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters:

*timeout* - Time in Milliseconds to wait for the task to stop.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1225), and `activemq::threads::DedicatedTaskRunner` (p. 1672).

**6.800.2.3** virtual void activemq::threads::TaskRunner::wakeup () [pure virtual]

Signal the **TaskRunner** (p. 3736) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3734) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1225), and **activemq::threads::DedicatedTaskRunner** (p. 1672).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

## 6.801 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

#include <src/main/decaf/internal/net/tcp/TcpSocket.h> Inheritance diagram for decaf::internal::net::tcp::TcpSocket:

### Public Member Functions

- **TcpSocket** () throw ( decaf::net::SocketException )  
*Construct a non-connected socket.*
- virtual ~**TcpSocket** ()  
*Releases the socket handle but not gracefully shut down the connection.*
- SocketHandle **getSocketHandle** ()  
*Gets the handle for the socket.*
- bool **isConnected** () const
- bool **isClosed** () const
- virtual std::string **getLocalAddress** () const  
*Gets the value of the local Inet address the **Socket** (p. 3503) is bound to if bound, otherwise return the **InetAddress** (p. 2016) ANY value "0.0.0.0".*  
**Returns:**  
*the local address bound to, or ANY.*
- virtual void **create** () throw ( decaf::io::IOException )  
*Creates the underlying platform **Socket** (p. 3503) data structures which allows for **Socket** (p. 3503) options to be applied.*  
*The created socket is in an unconnected state.*  
**Exceptions:**  
***IOException** if an I/O error occurs while attempting this operation.*
- virtual void **accept** (SocketImpl \*socket) throw ( decaf::io::IOException )
- virtual void **bind** (const std::string &ipaddress, int **port**) throw ( decaf::io::IOException )  
*Binds this **Socket** (p. 3503) instance to the local ip address and port number given.*  
**Parameters:**  
***ipaddress** The address of local ip to bind to.*  
***port** The port number on the host to bind to.*  
**Exceptions:**  
***IOException** if an I/O error occurs while attempting this operation.*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects this socket to the given host and port.*

**Parameters:**

**hostname** The name of the host to connect to, or IP address.  
**port** The port number on the host to connect to.  
**timeout** Time in milliseconds to wait for a connection, 0 indicates forever.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.  
**SocketTimeoutException** (p. 3542) if the connect call times out due to timeout being  
<sup>set.</sup>  
**IllegalArgumentException** if a parameter has an illegal value.

- virtual void **listen** (int backlog) throw ( decaf::io::IOException )

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

**Parameters:**

**backlog** The maximum length of the connection queue.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

- virtual decaf::io::InputStream \* **getInputStream** () throw ( decaf::io::IOException )

Gets the InputStream linked to this **Socket** (p. 3503).

**Returns:**

an InputStream pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

- virtual decaf::io::OutputStream \* **getOutputStream** () throw ( decaf::io::IOException )

Gets the OutputStream linked to this **Socket** (p. 3503).

**Returns:**

an OutputStream pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

- virtual int **available** () throw ( decaf::io::IOException )

Gets the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

**Returns:**

the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

- virtual void **close** () throw ( decaf::io::IOException )

Closes the socket, terminating any blocked reads or writes.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

- virtual void **shutdownInput** () throw ( decaf::io::IOException )

*Places the input stream for this socket at "end of stream".*

*Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3535) on the socket, the stream will return *EOF*.*

**Exceptions:**

***IOException** if an I/O error occurs while attempting this operation.*

- virtual void **shutdownOutput** () throw ( decaf::io::IOException )

*Disables the output stream for this socket.*

*For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3536) on the socket, the stream will throw an **IOException**.*

**Exceptions:**

***IOException** if an I/O error occurs while attempting this operation.*

- virtual int **getOption** (int option) const throw ( decaf::io::IOException )

*Gets the specified **Socket** (p. 3503) option.*

**Parameters:**

***option** The **Socket** (p. 3503) options whose value is to be retrieved.*

**Returns:**

*the value of the given socket option.*

**Exceptions:**

***IOException** if an I/O error occurs while performing this operation.*

- virtual void **setOption** (int option, int value) throw ( decaf::io::IOException )

*Sets the specified option on the **Socket** (p. 3503) if supported.*

**Parameters:**

***option** The **Socket** (p. 3503) option to set.*

***value** The value of the socket option to apply to the socket.*

**Exceptions:**

***IOException** if an I/O error occurs while performing this operation.*

- int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Reads the requested data from the **Socket** and write it into the passed in buffer.*

- void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Writes the specified data in the passed in buffer to the **Socket**.*

## Protected Member Functions

- void **checkResult** (apr\_status\_t value) const throw ( decaf::net::SocketException )

### 6.801.1 Detailed Description

Platform-independent implementation of the socket interface.

## 6.801.2 Constructor & Destructor Documentation

### 6.801.2.1 `decaf::internal::net::tcp::TcpSocket::TcpSocket () throw ( decaf::net::SocketException )`

Construct a non-connected socket.

#### Exceptions:

*SocketException* thrown if an error occurs while creating the Socket.

### 6.801.2.2 `virtual decaf::internal::net::tcp::TcpSocket::~~TcpSocket () [virtual]`

Releases the socket handle but not gracefully shut down the connection.

## 6.801.3 Member Function Documentation

### 6.801.3.1 `virtual void decaf::internal::net::tcp::TcpSocket::accept (SocketImpl * socket) throw ( decaf::io::IOException ) [virtual]`

### 6.801.3.2 `virtual int decaf::internal::net::tcp::TcpSocket::available () throw ( decaf::io::IOException ) [virtual]`

Gets the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

#### Returns:

the number of bytes that can be read from the **Socket** (p. 3503) without blocking.

#### Exceptions:

*IOException* if an I/O error occurs while attempting this operation.

Implements `decaf::net::SocketImpl` (p. 3531).

### 6.801.3.3 `virtual void decaf::internal::net::tcp::TcpSocket::bind (const std::string & ipaddress, int port) throw ( decaf::io::IOException ) [virtual]`

Binds this **Socket** (p. 3503) instance to the local ip address and port number given.

#### Parameters:

*ipaddress* The address of local ip to bind to.

*port* The port number on the host to bind to.

#### Exceptions:

*IOException* if an I/O error occurs while attempting this operation.

Implements `decaf::net::SocketImpl` (p. 3531).

**6.801.3.4** void decaf::internal::net::tcp::TcpSocket::checkResult (apr\_status\_t value) const throw ( decaf::net::SocketException ) [protected]

**6.801.3.5** virtual void decaf::internal::net::tcp::TcpSocket::close () throw ( decaf::io::IOException ) [virtual]

Closes the socket, terminating any blocked reads or writes.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implements decaf::net::SocketImpl (p. 3532).

**6.801.3.6** virtual void decaf::internal::net::tcp::TcpSocket::connect (const std::string & hostname, int port, int timeout) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Connects this socket to the given host and port.

**Parameters:**

*hostname* The name of the host to connect to, or IP address.

*port* The port number on the host to connect to.

*timeout* Time in milliseconds to wait for a connection, 0 indicates forever.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

*SocketTimeoutException* (p. 3542) if the connect call times out due to timeout being set.

*IllegalArgumentException* if a parameter has an illegal value.

Implements decaf::net::SocketImpl (p. 3532).

**6.801.3.7** virtual void decaf::internal::net::tcp::TcpSocket::create () throw ( decaf::io::IOException ) [virtual]

Creates the underlying platform **Socket** (p. 3503) data structures which allows for **Socket** (p. 3503) options to be applied.

The created socket is in an unconnected state.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implements decaf::net::SocketImpl (p. 3532).

**6.801.3.8** `virtual decaf::io::InputStream* decaf::internal::net::tcp::TcpSocket::getInputStream () throw ( decaf::io::IOException ) [virtual]`

Gets the `InputStream` linked to this **Socket** (p. 3503).

**Returns:**

an `InputStream` pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

***IOException*** if an I/O error occurs while attempting this operation.

Implements `decaf::net::SocketImpl` (p. 3533).

**6.801.3.9** `virtual std::string decaf::internal::net::tcp::TcpSocket::getLocalAddress () const [virtual]`

Gets the value of the local `Inet` address the **Socket** (p. 3503) is bound to if bound, otherwise return the **InetAddress** (p. 2016) ANY value "0.0.0.0".

**Returns:**

the local address bound to, or ANY.

Implements `decaf::net::SocketImpl` (p. 3533).

**6.801.3.10** `virtual int decaf::internal::net::tcp::TcpSocket::getOption (int option) const throw ( decaf::io::IOException ) [virtual]`

Gets the specified **Socket** (p. 3503) option.

**Parameters:**

***option*** The **Socket** (p. 3503) options whose value is to be retrieved.

**Returns:**

the value of the given socket option.

**Exceptions:**

***IOException*** if an I/O error occurs while performing this operation.

Implements `decaf::net::SocketImpl` (p. 3534).

**6.801.3.11** `virtual decaf::io::OutputStream* decaf::internal::net::tcp::TcpSocket::getOutputStream () throw ( decaf::io::IOException ) [virtual]`

Gets the `OutputStream` linked to this **Socket** (p. 3503).



**Returns:**

an OutputStream pointer owned by the **Socket** (p. 3503) object.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3534).

**6.801.3.12 SocketHandle decaf::internal::net::tcp::TcpSocket::getSocketHandle ()**  
[inline]

Gets the handle for the socket.

**Returns:**

SocketHabler for this Socket, can be NULL

**6.801.3.13 bool decaf::internal::net::tcp::TcpSocket::isClosed () const** [inline]**Returns:**

true if the close method has been called on this Socket.

**6.801.3.14 bool decaf::internal::net::tcp::TcpSocket::isConnected () const** [inline]**Returns:**

true if the socketHandle is not in a disconnected state.

**6.801.3.15 virtual void decaf::internal::net::tcp::TcpSocket::listen (int *backlog*)**  
**throw ( decaf::io::IOException )** [virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

**Parameters:**

*backlog* The maximum length of the connection queue.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3534).

**6.801.3.16** `int decaf::internal::net::tcp::TcpSocket::read (unsigned char * buffer,  
int size, int offset, int length) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

Reads the requested data from the Socket and write it into the passed in buffer.

**Parameters:**

*buffer* The buffer to read into  
*size* The size of the specified buffer  
*offset* The offset into the buffer where reading should start filling.  
*length* The number of bytes past offset to fill with data.

**Returns:**

the actual number of bytes read or -1 if at EOF.

**Exceptions:**

*IOException* if an I/O error occurs during the read.  
*NullPointerException* if buffer is Null.  
*IndexOutOfBoundsException* if offset + length is greater than buffer size.

**6.801.3.17** `virtual void decaf::internal::net::tcp::TcpSocket::setOption (int option,  
int value) throw ( decaf::io::IOException ) [virtual]`

Sets the specified option on the **Socket** (p. 3503) if supported.

**Parameters:**

*option* The **Socket** (p. 3503) option to set.  
*value* The value of the socket option to apply to the socket.

**Exceptions:**

*IOException* if an I/O error occurs while performing this operation.

Implements **decaf::net::SocketImpl** (p. 3535).

**6.801.3.18** `virtual void decaf::internal::net::tcp::TcpSocket::shutdownInput () throw  
( decaf::io::IOException ) [virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3535) on the socket, the stream will return EOF.

**Exceptions:**

*IOException* if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3535).

**6.801.3.19 virtual void decaf::internal::net::tcp::TcpSocket::shutdownOutput ()  
throw ( decaf::io::IOException ) [virtual]**

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3536) on the socket, the stream will throw an **IOException**.

**Exceptions:**

**IOException** if an I/O error occurs while attempting this operation.

Implements **decaf::net::SocketImpl** (p. 3536).

**6.801.3.20 void decaf::internal::net::tcp::TcpSocket::write (const unsigned char \*  
buffer, int size, int offset, int length) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )**

Writes the specified data in the passed in buffer to the Socket.

**Parameters:**

**buffer** The buffer to write to the socket.

**size** The size of the specified buffer.

**offset** The offset into the buffer where the data to write starts at.

**length** The number of bytes past offset to write.

**Exceptions:**

**IOException** if an I/O error occurs during the write.

**NullPointerException** if buffer is Null.

**IndexOutOfBoundsException** if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocket.h**

## 6.802 decaf::internal::net::tcp::TcpSocketInputStream Class Reference

Input stream for performing reads on a socket.

#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h> Inheritance diagram for decaf::internal::net::tcp::TcpSocketInputStream:

### Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** \*socket)  
*Create a new InputStream to use for reading from the TCP/IP socket.*
- virtual **~TcpSocketInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.*  
**Returns:**  
*the number of bytes available on this input stream.*  
**Exceptions:**  
*IOException (p. 2142) if an I/O error occurs.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Close - does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Not supported.*

### Protected Member Functions

- virtual int **doReadByte** () throw ( io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

#### 6.802.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

**Since:**

1.0

## 6.802.2 Constructor & Destructor Documentation

### 6.802.2.1 decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream (TcpSocket \* *socket*)

Create a new `InputStream` to use for reading from the TCP/IP socket.

#### Parameters:

*socket* The parent `SocketImpl` for this stream.

### 6.802.2.2 virtual decaf::internal::net::tcp::TcpSocketInputStream::~TcpSocketInputStream () [virtual]

## 6.802.3 Member Function Documentation

### 6.802.3.1 virtual int decaf::internal::net::tcp::TcpSocketInputStream::available () const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns:

the number of bytes available on this input stream.

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

Reimplemented from `decaf::io::InputStream` (p. 2045).

### 6.802.3.2 virtual void decaf::internal::net::tcp::TcpSocketInputStream::close () throw ( decaf::io::IOException ) [virtual]

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Closes the `InputStream` (p. 2043) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::InputStream` (p. 2045).

**6.802.3.3** `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadArrayBounded (unsigned char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from `decaf::io::InputStream` (p. 2046).

**6.802.3.4** `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte () throw ( io::IOException )` [protected, virtual]

Implements `decaf::io::InputStream` (p. 2046).

**6.802.3.5** `virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Not supported. Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of `InputStream` (p. 2043) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters:

*num* The number of bytes to skip.

#### Returns:

total bytes skipped

#### Exceptions:

*IOException* (p. 2142) if an I/O error occurs.

*UnsupportedOperationException* if the concrete stream class does not support skipping bytes.

Reimplemented from `decaf::io::InputStream` (p. 2050).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/tcp/TcpSocketInputStream.h`

## 6.803 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference

Output stream for performing write operations on a socket.

#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>Inheritance diagram for decaf::internal::net::tcp::TcpSocketOutputStream:

### Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** \*socket)  
*Create a new instance of a Socket OutputStream class.*
- virtual ~**TcpSocketOutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*  
**Exceptions:**  
***IOException** (p. 2142) if an error occurs while closing.  
The default implementation of this method does nothing.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.803.1 Detailed Description

Output stream for performing write operations on a socket.

**Since:**

1.0

### 6.803.2 Constructor & Destructor Documentation

#### 6.803.2.1 decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream (**TcpSocket** \* socket)

Create a new instance of a Socket OutputStream class.

**Parameters:**

*socket* The socket to use to write out the data.

**6.803.2.2**    **virtual**  
**decaf::internal::net::tcp::TcpSocketOutputStream::~~TcpSocketOutputStream**  
**()**    [virtual]

### 6.803.3    Member Function Documentation

**6.803.3.1**    **virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ()**  
**throw ( decaf::io::IOException )**    [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions:

*IOException* (p. 2142) if an error occurs while closing.

The default implementation of this method does nothing. The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2909).

**6.803.3.2**    **virtual void de-**  
**caf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded**  
**(const unsigned char \* *buffer*, int *size*, int *offset*, int *length*) throw (**  
**decaf::io::IOException, decaf::lang::exceptions::NullPointerException,**  
**decaf::lang::exceptions::IndexOutOfBoundsException )**    [protected,  
virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2909).

**6.803.3.3**    **virtual void de-**  
**caf::internal::net::tcp::TcpSocketOutputStream::doWriteByte (unsigned**  
**char *c*) throw ( decaf::io::IOException )**    [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2910).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h`



## 6.804 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based **transport** (p. 97) filter, this **transport** (p. 97) is meant to wrap an instance of an **IOTransport** (p. 2145).

#include <src/main/activemq/transport/tcp/TcpTransport.h> Inheritance diagram for activemq::transport::tcp::TcpTransport:

### Public Member Functions

- **TcpTransport** (const **Pointer**< **Transport** > &next)  
*Creates a new instance of a **TcpTransport** (p. 3752), the **transport** (p. 97) is left unconnected and is in a unusable **state** (p. 95) until the connect method is called.*
- virtual ~**TcpTransport** ()
- void **connect** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties)  
*Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.*
- virtual void **close** () throw ( **decaf::io::IOException** )  
*Delegates to the superclass and then closes the socket.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3883) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3883) been shutdown and no longer usable.*

### Protected Member Functions

- virtual **decaf::net::Socket** \* **createSocket** ()  
*Create an unconnected Socket instance to be used by the **transport** (p. 97) to communicate with the broker.*
- virtual void **configureSocket** (**decaf::net::Socket** \*socket, const **decaf::util::Properties** &properties)  
*Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.*

### 6.804.1 Detailed Description

Implements a TCP/IP based **transport** (p. 97) filter, this **transport** (p. 97) is meant to wrap an instance of an **IOTransport** (p. 2145). The lower level **transport** (p. 97) should take care of managing stream reads and writes.

### 6.804.2 Constructor & Destructor Documentation

#### 6.804.2.1 **activemq::transport::tcp::TcpTransport::TcpTransport** (const **Pointer**<**Transport** > & *next*)

Creates a new instance of a **TcpTransport** (p. 3752), the **transport** (p. 97) is left unconnected and is in a unusable **state** (p. 95) until the connect method is called.

##### Parameters:

*next* The next **transport** (p. 97) in the chain

#### 6.804.2.2 **virtual activemq::transport::tcp::TcpTransport::~~TcpTransport** () [virtual]

### 6.804.3 Member Function Documentation

#### 6.804.3.1 **virtual void activemq::transport::tcp::TcpTransport::close** () throw (**decaf::io::IOException**) [virtual]

Delegates to the superclass and then closes the socket.

##### Exceptions:

**IOException** if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3893).

#### 6.804.3.2 **virtual void activemq::transport::tcp::TcpTransport::configureSocket** (**decaf::net::Socket** \* *socket*, const **decaf::util::Properties** & *properties*) [protected, virtual]

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server. Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

##### Parameters:

*socket* The Socket instance to configure using options from the given Properties.

##### Exceptions:

**NullPointerException** if the Socket instance is null.

**IllegalArgumentException** if the socket instance is not handled by the class.

**SocketException** if there is an error while setting one of the Socket options.

### 6.804.3.3 void activemq::transport::tcp::TcpTransport::connect (const decaf::net::URI & *uri*, const decaf::util::Properties & *properties*)

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in. The Socket is configured using parameters in the properties that are passed to this method.

#### Parameters:

***uri*** The URI that the **Transport** (p. 3883) is to connect to once initialized.

***properties*** The Properties that have been parsed from the URI or from configuration files.

### 6.804.3.4 virtual decaf::net::Socket\* activemq::transport::tcp::TcpTransport::createSocket () [protected, virtual]

Create an unconnected Socket instance to be used by the **transport** (p. 97) to communicate with the broker.

#### Returns:

a newly created unconnected Socket instance.

#### Exceptions:

***IOException*** if there is an error while creating the unconnected Socket.

Reimplemented in **activemq::transport::tcp::SslTransport** (p. 3575).

### 6.804.3.5 virtual bool activemq::transport::tcp::TcpTransport::isClosed () const [inline, virtual]

Has the **Transport** (p. 3883) been shutdown and no longer usable.

#### Returns:

true if the **Transport** (p. 3883)

Reimplemented from **activemq::transport::TransportFilter** (p. 3894).

### 6.804.3.6 virtual bool activemq::transport::tcp::TcpTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 3883) Connected to its Broker.

#### Returns:

true if a connection has been made.

Reimplemented from **activemq::transport::TransportFilter** (p. 3894).

**6.804.3.7** `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ()`  
`const [inline, virtual]`

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns:**

true if the **Transport** (p. 3883) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 3895).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

## 6.805 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 3752).

#include <src/main/activemq/transport/tcp/TcpTransportFactory.h> Inheritance diagram for activemq::transport::tcp::TcpTransportFactory:

### Public Member Functions

- virtual **~TcpTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )

*Creates a fully configured **Transport** (p. 3883) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )

*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

### Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw ( exceptions::ActiveMQException )

*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

### 6.805.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 3752).

## 6.805.2 Constructor & Destructor Documentation

**6.805.2.1** `virtual  
activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory ()  
[inline, virtual]`

## 6.805.3 Member Function Documentation

**6.805.3.1** `virtual Pointer<Transport> ac-  
tivismq::transport::tcp::TcpTransportFactory::create (const  
decaf::net::URI & location) throw ( exceptions::ActiveMQException )  
[virtual]`

Creates a fully configured **Transport** (p.3883) instance which could be a chain of filters and transports.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements `activemq::transport::TransportFactory` (p.3889).

**6.805.3.2** `virtual Pointer<Transport> ac-  
tivismq::transport::tcp::TcpTransportFactory::createComposite (const  
decaf::net::URI & location) throw ( exceptions::ActiveMQException )  
[virtual]`

Creates a slimed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.

### Parameters:

*location* - URI location to connect to plus any properties to assign.

### Exceptions:

*ActiveMQException* if an error occurs

Implements `activemq::transport::TransportFactory` (p.3890).

**6.805.3.3** `virtual Pointer<Transport> ac-  
tivismq::transport::tcp::TcpTransportFactory::doCreateComposite (const  
decaf::net::URI & location, const Pointer< wireformat::WireFormat  
> & wireFormat, const decaf::util::Properties & properties) throw (   
exceptions::ActiveMQException ) [protected, virtual]`

Creates a slimed down **Transport** (p.3883) instance which can be used in composite **transport** (p.97) instances.

**Parameters:**

*location* - URI location to connect to.

*wireFormat* - the assigned WireFormat for the new **Transport** (p. 3883).

*properties* - Properties to apply to the **transport** (p. 97).

**Returns:**

new Pointer to a **TcpTransport** (p. 3752).

**Exceptions:**

*ActiveMQException* if an error occurs

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 3576).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransportFactory.h`

## 6.806 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 3148) based **Destination** (p. 1723).

#include <src/main/cms/TemporaryQueue.h> Inheritance diagram for cms::TemporaryQueue:

### Public Member Functions

- virtual **~TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw ( CMSEException )  
*Gets the name of this queue.*
- virtual void **destroy** ()=0 throw ( CMSEException )  
*Destroy's the Temporary **Destination** (p. 1723) at the Provider.*

### 6.806.1 Detailed Description

Defines a Temporary **Queue** (p. 3148) based **Destination** (p. 1723). A **TemporaryQueue** (p. 3759) is a special type of **Queue** (p. 3148) **Destination** (p. 1723) that can only be consumed from the **Connection** (p. 1262) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 3759) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1262) that created it.

Since:

1.0

### 6.806.2 Constructor & Destructor Documentation

**6.806.2.1** virtual cms::TemporaryQueue::~~TemporaryQueue () [inline, virtual]

### 6.806.3 Member Function Documentation

**6.806.3.1** virtual void cms::TemporaryQueue::destroy () throw ( CMSEException )  
 [pure virtual]

Destroy's the Temporary **Destination** (p. 1723) at the Provider.

Exceptions:

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 606).



**6.806.3.2** virtual std::string cms::TemporaryQueue::getQueueName () const throw ( CMSEException ) [pure virtual]

Gets the name of this queue.

**Returns:**

The queue name.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 607).

The documentation for this class was generated from the following file:

- src/main/cms/TemporaryQueue.h

## 6.807 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3817) based **Destination** (p. 1723).

#include <src/main/cms/TemporaryTopic.h> Inheritance diagram for cms::TemporaryTopic:

### Public Member Functions

- virtual **~TemporaryTopic** ()
- virtual std::string **getTopicName** () const =0 throw ( CMSEException )  
*Gets the name of this topic.*
- virtual void **destroy** ()=0 throw ( CMSEException )  
*Destroy's the Temporary **Destination** (p. 1723) at the Provider.*

### 6.807.1 Detailed Description

Defines a Temporary **Topic** (p. 3817) based **Destination** (p. 1723). A **TemporaryTopic** (p. 3761) is a special type of **Topic** (p. 3817) **Destination** (p. 1723) that can only be consumed from the **Connection** (p. 1262) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 3761) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1262) that created it.

Since:

1.0

### 6.807.2 Constructor & Destructor Documentation

**6.807.2.1** virtual cms::TemporaryTopic::~~TemporaryTopic () [inline, virtual]

### 6.807.3 Member Function Documentation

**6.807.3.1** virtual void cms::TemporaryTopic::destroy () throw ( CMSEException )  
[pure virtual]

Destroy's the Temporary **Destination** (p. 1723) at the Provider.

Exceptions:

*CMSEException* (p. 1160)

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 635).

**6.807.3.2** `virtual std::string cms::TemporaryTopic::getTopicName () const throw ( CMSException ) [pure virtual]`

Gets the name of this topic.

**Returns:**

The topic name.

**Exceptions:**

*CMSException* (p. 1160) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 636).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryTopic.h`

## 6.808 cms::TextMessage Class Reference

Interface for a text message.

#include <src/main/cms/TextMessage.h> Inheritance diagram for cms::TextMessage:

### Public Member Functions

- virtual `~TextMessage ()`
- virtual `std::string getText () const` `=0` throw ( cms::CMSEException )  
*Gets the message character buffer.*
- virtual void `setText (const char *msg)=0` throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*
- virtual void `setText (const std::string &msg)=0` throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets the message contents.*

### 6.808.1 Detailed Description

Interface for a text message. A **TextMessage** (p. 3763) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 3763) is received in Read-Only mode, any attempt to write to the **Message** (p. 2534) will result in a `MessageNotWritableException` being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since:

1.0

### 6.808.2 Constructor & Destructor Documentation

**6.808.2.1** virtual cms::TextMessage::~TextMessage () [inline, virtual]

### 6.808.3 Member Function Documentation

**6.808.3.1** virtual std::string cms::TextMessage::getText () const throw ( cms::CMSEException ) [pure virtual]

Gets the message character buffer.

Returns:

The message character buffer.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 665).

**6.808.3.2** `virtual void cms::TextMessage::setText (const std::string & msg) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets the message contents.

**Parameters:**

*msg* The message buffer.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 665).

**6.808.3.3** `virtual void cms::TextMessage::setText (const char * msg) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

**Parameters:**

*msg* The message buffer.

**Exceptions:**

*CMSEException* (p. 1160) - if an internal error occurs.

*MessageNotWriteableException* (p. 2724) - if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 665).

The documentation for this class was generated from the following file:

- `src/main/cms/TextMessage.h`

## 6.809 decaf::lang::Thread Class Reference

A **Thread** (p. 3765) is a concurrent unit of execution.

#include <src/main/decaf/lang/Thread.h> Inheritance diagram for decaf::lang::Thread:

### Data Structures

- class **UncaughtExceptionHandler**

*Interface for handlers invoked when a **Thread** (p. 3765) abruptly terminates due to an uncaught exception.*

### Public Types

- enum **State** {  
**NEW** = 0, **RUNNABLE** = 1, **BLOCKED** = 2, **WAITING** = 3,  
**TIMED\_WAITING** = 4, **SLEEPING** = 5, **TERMINATED** = 6 }

*Represents the various states that the **Thread** (p. 3765) can be in during its lifetime.*

### Public Member Functions

- **Thread** ()  
*Constructs a new **Thread** (p. 3765).*
- **Thread** (**Runnable** \*task)  
*Constructs a new **Thread** (p. 3765) with the given target **Runnable** (p. 3325) task.*
- **Thread** (const std::string &name)  
*Constructs a new **Thread** (p. 3765) with the given name.*
- **Thread** (**Runnable** \*task, const std::string &name)  
*Constructs a new **Thread** (p. 3765) with the given target **Runnable** (p. 3325) task and name.*
- virtual ~**Thread** ()
- virtual void **start** () throw ( decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException )  
*Creates a system thread and starts it in a joinable mode.*
- virtual void **join** () throw ( decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3765) to wait until the thread exits.*
- virtual void **join** (long long millisecs) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3765) to wait until the thread exits.*

- virtual void **join** (long long millisecs, unsigned int nanos) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3765) to wait until the thread exits.*
- virtual void **run** ()  
*Default implementation of the run method - does nothing.*
- std::string **getName** () const  
*Returns the Thread's assigned name.*
- void **setName** (const std::string &name)  
*Sets the name of the **Thread** (p. 3765) to the new Name given by the argument name.*
- int **getPriority** () const  
*Gets the currently set priority for this **Thread** (p. 3765).*
- void **setPriority** (int value) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Sets the current Thread's priority to the newly specified value.*
- const **UncaughtExceptionHandler** \* **getUncaughtExceptionHandler** () const  
*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*
- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** \*handler)  
*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*
- std::string **toString** () const  
*Returns a string that describes the **Thread** (p. 3765).*
- bool **isAlive** () const  
*Returns true if the **Thread** (p. 3765) is alive, meaning it has been started and has not yet died.*
- **Thread::State** **getState** () const  
*Returns the currently set State of this **Thread** (p. 3765).*

## Static Public Member Functions

- static void **sleep** (long long millisecs) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )  
*Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.*
- static void **sleep** (long long millisecs, unsigned int nanos) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )  
*Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.*

- static void **yield** ()  
*Causes the currently executing thread object to temporarily pause and allow other threads to execute.*
- static long long **getId** ()  
*Obtains the **Thread** (p. 3765) Id of the current thread.*
- static **Thread** \* **currentThread** ()  
*Returns a pointer to the currently executing thread object.*

## Static Public Attributes

- static const int **MIN\_PRIORITY** = 1  
*The minimum priority that a thread can have.*
- static const int **NORM\_PRIORITY** = 5  
*The default priority that a thread is given at create time.*
- static const int **MAX\_PRIORITY** = 10  
*The maximum priority that a thread can have.*

## Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

### 6.809.1 Detailed Description

A **Thread** (p. 3765) is a concurrent unit of execution. It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 3765) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional Threads for specific purposes.

Threads in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 3765) execute application **code** (p. 1183). One is providing a new class that extends **Thread** (p. 3765) and overriding its **run()** (p. 3771) method. The other is providing a new **Thread** (p. 3765) instance with a **Runnable** (p. 3325) object during its creation. In both cases, the **start()** (p. 3772) method must be called to actually execute the new **Thread** (p. 3765).

Each **Thread** (p. 3765) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 3765) gets. It can be set using the **setPriority(int)** (p. 3771) method. A **Thread** (p. 3765) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

See also:

**decaf.lang.ThreadGroup** (p. 3776)



Since:

1.0

## 6.809.2 Member Enumeration Documentation

### 6.809.2.1 enum decaf::lang::Thread::State

Represents the various states that the **Thread** (p. 3765) can be in during its lifetime.

Enumerator:

**NEW** Before a **Thread** (p. 3765) is started it exists in this State.

**RUNNABLE** While a **Thread** (p. 3765) is running and is not blocked it is in this State.

**BLOCKED** A **Thread** (p. 3765) that is waiting to acquire a lock is in this state.

**WAITING** A **Thread** (p. 3765) that is waiting for another **Thread** (p. 3765) to perform an action is in this state.

**TIMED\_WAITING** A **Thread** (p. 3765) that is waiting for another **Thread** (p. 3765) to perform an action up to a specified time interval is in this state.

**SLEEPING** A **Thread** (p. 3765) that is blocked in a Sleep call is in this state.

**TERMINATED** A **Thread** (p. 3765) whose run method has exited is in this state.

## 6.809.3 Constructor & Destructor Documentation

### 6.809.3.1 decaf::lang::Thread::Thread ()

Constructs a new **Thread** (p. 3765). This constructor has the same effect as Thread( NULL, NULL, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

### 6.809.3.2 decaf::lang::Thread::Thread (Runnable \* task)

Constructs a new **Thread** (p. 3765) with the given target **Runnable** (p. 3325) task. This constructor has the same effect as Thread( NULL, task, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

Parameters:

**task** the **Runnable** (p. 3325) that this thread manages, if the task is NULL the Thread's run method is used instead.

### 6.809.3.3 decaf::lang::Thread::Thread (const std::string & name)

Constructs a new **Thread** (p. 3765) with the given name. This constructor has the same effect as Thread( NULL, NULL, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

**Parameters:**

*name* the name to assign to this **Thread** (p. 3765).

**6.809.3.4 decaf::lang::Thread::Thread (Runnable \* *task*, const std::string & *name*)**

Constructs a new **Thread** (p. 3765) with the given target **Runnable** (p. 3325) task and name. This constructor has the same effect as Thread( NULL, task, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

**Parameters:**

*task* the **Runnable** (p. 3325) that this thread manages, if the task is NULL the Thread's run method is used instead.

*name* the name to assign to this **Thread** (p. 3765).

**6.809.3.5 virtual decaf::lang::Thread::~~Thread () [virtual]****6.809.4 Member Function Documentation****6.809.4.1 static Thread\* decaf::lang::Thread::currentThread () [static]**

Returns a pointer to the currently executing thread object.

**Returns:**

**Pointer** (p. 2946) to the **Thread** (p. 3765) object representing the currently running **Thread** (p. 3765).

**6.809.4.2 static long long decaf::lang::Thread::getId () [static]**

Obtains the **Thread** (p. 3765) Id of the current thread.

**Returns:**

**Thread** (p. 3765) Id

**6.809.4.3 std::string decaf::lang::Thread::getName () const**

Returns the Thread's assigned name.

**Returns:**

the Name of the **Thread** (p. 3765).

**6.809.4.4 int decaf::lang::Thread::getPriority () const**

Gets the currently set priority for this **Thread** (p. 3765).

**Returns:**

an int value representing the Thread's current priority.

**6.809.4.5 Thread::State decaf::lang::Thread::getState () const**

Returns the currently set State of this **Thread** (p. 3765).

**Returns:**

the Thread's current state.

**6.809.4.6 const UncaughtExceptionHandler\* decaf::lang::Thread::getUncaughtExceptionHandler () const**

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

**Returns:**

a pointer to the set **UncaughtExceptionHandler** (p. 3906).

**6.809.4.7 bool decaf::lang::Thread::isAlive () const**

Returns true if the **Thread** (p. 3765) is alive, meaning it has been started and has not yet died.

**Returns:**

true if the thread is alive.

**6.809.4.8 virtual void decaf::lang::Thread::join (long long *millisecs*, unsigned int *nanos*) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException ) [virtual]**

Forces the Current **Thread** (p. 3765) to wait until the thread exits.

**Parameters:**

*millisecs* the time in Milliseconds before the thread resumes

*nanos* 0-999999 extra nanoseconds to sleep.

**Exceptions:**

***IllegalArgumentException*** if the nanoseconds parameter is out of range or the milliseconds paramter is negative.

***InterruptedException*** if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

**6.809.4.9 virtual void decaf::lang::Thread::join (long long *millisecs*) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException ) [virtual]**

Forces the Current **Thread** (p. 3765) to wait until the thread exits.

**Parameters:**

*milliseconds* the time in Milliseconds before the thread resumes

**Exceptions:**

*IllegalArgumentException* if the milliseconds parameter is negative.

*InterruptedException* if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

**6.809.4.10** `virtual void decaf::lang::Thread::join () throw ( decaf::lang::exceptions::InterruptedException ) [virtual]`

Forces the Current **Thread** (p. 3765) to wait until the thread exits.

**Exceptions:**

*InterruptedException* if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

**6.809.4.11** `virtual void decaf::lang::Thread::run () [virtual]`

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 3325).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 2123), and **decaf::util::concurrent::PooledThread** (p. 2969).

**6.809.4.12** `void decaf::lang::Thread::setName (const std::string & name)`

Sets the name of the **Thread** (p. 3765) to the new Name given by the argument *name*. name the new name of the **Thread** (p. 3765).

**6.809.4.13** `void decaf::lang::Thread::setPriority (int value) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Sets the current Thread's priority to the newly specified value. The given value must be within the range **Thread::MIN\_PRIORITY** (p. 3773) and **Thread::MAX\_PRIORITY** (p. 3773).

**Parameters:**

*value* the new priority value to assign to this **Thread** (p. 3765).

**Exceptions:**

*IllegalArgumentException* if the value is out of range.

**6.809.4.14 void decaf::lang::Thread::setUncaughtExceptionHandler  
(UncaughtExceptionHandler \* *handler*)**

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

**Parameters:**

*handler* the UncaughtExceptionHandler to invoke when the **Thread** (p. 3765) terminates due to an uncaught exception.

**6.809.4.15 static void decaf::lang::Thread::sleep (long long *milliseconds*, unsigned  
int *nanos*) throw ( lang::exceptions::InterruptedException,  
lang::exceptions::IllegalArgumentException ) [static]**

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers. Note that this method is a static method that applies to the calling thread and not to the thread object.

**Parameters:**

*milliseconds* time in milliseconds to halt execution.

*nanos* 0-999999 extra nanoseconds to sleep.

**Exceptions:**

**IllegalArgumentException** if the nanoseconds parameter is out of range or the milliseconds parameter is negative.

**InterruptedException** if the **Thread** (p. 3765) was interrupted while sleeping.

**6.809.4.16 static void decaf::lang::Thread::sleep (long long  
*milliseconds*) throw ( lang::exceptions::InterruptedException,  
lang::exceptions::IllegalArgumentException ) [static]**

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers. Note that this method is a static method that applies to the calling thread and not to the thread object.

**Parameters:**

*milliseconds* time in milliseconds to halt execution.

**Exceptions:**

**IllegalArgumentException** if the milliseconds parameter is negative.

**InterruptedException** if the **Thread** (p. 3765) was interrupted while sleeping.

**6.809.4.17 virtual void decaf::lang::Thread::start () throw (  
decaf::lang::exceptions::IllegalThreadStateException,  
decaf::lang::exceptions::RuntimeException ) [virtual]**

Creates a system thread and starts it in a joinable mode. Upon creation, the **run()** (p. 3771) method of either this object or the provided **Runnable** (p. 3325) object will be invoked in the context of this thread.

**Exceptions:**

***IllegalThreadStateException*** if the thread has already been started.

***RuntimeException*** if the **Thread** (p. 3765) cannot be created for some reason.

**6.809.4.18** `std::string decaf::lang::Thread::toString () const`

Returns a string that describes the **Thread** (p. 3765).

**Returns:**

string describing the **Thread** (p. 3765).

**6.809.4.19** `static void decaf::lang::Thread::yield () [static]`

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

**6.809.5 Friends And Related Function Documentation****6.809.5.1** `friend class decaf::lang::Runtime [friend]`**6.809.5.2** `friend class decaf::util::concurrent::locks::LockSupport [friend]`**6.809.6 Field Documentation****6.809.6.1** `const int decaf::lang::Thread::MAX_PRIORITY = 10 [static]`

The maximum priority that a thread can have.

**6.809.6.2** `const int decaf::lang::Thread::MIN_PRIORITY = 1 [static]`

The minimum priority that a thread can have.

**6.809.6.3** `const int decaf::lang::Thread::NORM_PRIORITY = 5 [static]`

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Thread.h`

## 6.810 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 3774)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

### Public Member Functions

- virtual `~ThreadFactory()`
- virtual `decaf::lang::Thread * newThread (decaf::lang::Runnable *r)=0`  
*Constructs a new Thread.*

### 6.810.1 Detailed Description

public interface **ThreadFactory** (p. 3774) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 3774) { public: Thread* newThread(  
Runnable* r ) { return new Thread(r); } }
```

The `Executors.defaultThreadFactory()` method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since:

1.0

### 6.810.2 Constructor & Destructor Documentation

**6.810.2.1** virtual `decaf::util::concurrent::ThreadFactory::~~ThreadFactory()`  
[inline, virtual]

### 6.810.3 Member Function Documentation

**6.810.3.1** virtual `decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread (decaf::lang::Runnable * r)` [pure virtual]

Constructs a new Thread. Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

**Parameters:**

*r* A pointer to a Runnable instance to be executed by new Thread instance returned.

**Returns:**

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadFactory.h`



## 6.811 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

### Public Member Functions

- **ThreadGroup** ()
- virtual **~ThreadGroup** ()

#### 6.811.1 Detailed Description

Since:

1.0

#### 6.811.2 Constructor & Destructor Documentation

**6.811.2.1 decaf::lang::ThreadGroup::ThreadGroup ()**

**6.811.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup ()** [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ThreadGroup.h`

## 6.812 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

#include <src/main/decaf/util/concurrent/ThreadPool.h> Inheritance diagram for decaf::util::concurrent::ThreadPool:

### Public Types

- typedef std::pair< lang::Runnable \*, TaskListener \* > Task

### Public Member Functions

- **ThreadPool** ()
- virtual **~ThreadPool** ()
- virtual void **queueTask** (Task task) throw ( lang::Exception )  
*Queue (p. 3149) a task to be completed by one of the Pooled Threads.*
- virtual **Task deQueueTask** () throw ( lang::Exception )  
*DeQueue a task to be completed by one of the Pooled Threads.*
- virtual std::size\_t **getPoolSize** () const  
*Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.*
- virtual std::size\_t **getBacklog** () const  
*Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.*
- virtual void **reserve** (std::size\_t size)  
*Ensures that there is at least the specified number of Threads allocated to the pool.*
- virtual std::size\_t **getMaxThreads** () const  
*Get the Max Number of Threads this Pool can contain.*
- virtual void **setMaxThreads** (std::size\_t maxThreads)  
*Sets the Max number of threads this pool can contain.*
- virtual std::size\_t **getBlockSize** () const  
*Gets the Max number of threads that can be allocated at a time when new threads are needed.*
- virtual void **setBlockSize** (std::size\_t blockSize)  
*Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.*
- virtual std::size\_t **getFreeThreadCount** () const  
*Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.*

- virtual void **onTaskStarted** (**PooledThread** \*thread)

*Called by a pooled thread when it is about to begin executing a new task.*

- virtual void **onTaskCompleted** (**PooledThread** \*thread)

*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.*

- virtual void **onTaskException** (**PooledThread** \*thread, **lang::Exception** &ex)

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2968) is now no longer running.*

## Static Public Member Functions

- static **ThreadPool** \* **getInstance** ()

*Return the one and only Thread Pool instance.*

## Static Public Attributes

- static const size\_t **DEFAULT\_\_MAX\_\_POOL\_\_SIZE** = 10
- static const size\_t **DEFAULT\_\_MAX\_\_BLOCK\_\_SIZE** = 3

### 6.812.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the **Runnable** interface and one of the worker threads will executing it in its thread context.

## 6.812.2 Member Typedef Documentation

**6.812.2.1** `typedef std::pair<lang::Runnable*, TaskListener*>  
decaf::util::concurrent::ThreadPool::Task`

## 6.812.3 Constructor & Destructor Documentation

**6.812.3.1** `decaf::util::concurrent::ThreadPool::ThreadPool ()`

**6.812.3.2** `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool ()` [virtual]

## 6.812.4 Member Function Documentation

**6.812.4.1** `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask () throw (  
lang::Exception )` [virtual]

DeQueue a task to be completed by one of the Pooled Threads. A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p.3777) don't use this, only the **PooledThread** (p.2968) objects owned by this **ThreadPool** (p.3777).

### Returns:

object that derives from Runnable

### Exceptions:

*ActiveMQException*

**6.812.4.2** `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ()  
const` [inline, virtual]

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

### Returns:

number of outstanding tasks.

**6.812.4.3** `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize ()  
const` [inline, virtual]

Gets the Max number of threads that can be allocated at a time when new threads are needed.

### Returns:

max Thread Block Size

**6.812.4.4** `virtual std::size_t de-  
caf::util::concurrent::ThreadPool::getFreeThreadCount ()  
const` [inline, virtual]

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable. This value could change immediately after calling as Threads

could finish right after and be available again. This is informational only.

**Returns:**

total free threads

**6.812.4.5 static ThreadPool\* decaf::util::concurrent::ThreadPool::getInstance ()**  
[static]

Return the one and only Thread Pool instance.

**Returns:**

The Thread Pool Pointer

**6.812.4.6 virtual std::size\_t decaf::util::concurrent::ThreadPool::getMaxThreads ()**  
const [inline, virtual]

Get the Max Number of Threads this Pool can contain.

**Returns:**

max size

**6.812.4.7 virtual std::size\_t decaf::util::concurrent::ThreadPool::getPoolSize ()**  
const [inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

**Returns:**

integer number of threads in existence.

**6.812.4.8 virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted**  
(PooledThread \* *thread*) [virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

**Parameters:**

*thread* Pointer the the Pooled Thread that is making this call.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2971).

**6.812.4.9 virtual void decaf::util::concurrent::ThreadPool::onTaskException**  
(PooledThread \* *thread*, lang::Exception & *ex*) [virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2968) is now no longer running.

**Parameters:**

*thread* Pointer to the Pooled Thread that is making this call

*ex* The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2972).

#### 6.812.4.10 **virtual void decaf::util::concurrent::ThreadPool::onTaskStarted** (PooledThread \* *thread*) [virtual]

Called by a pooled thread when it is about to begin executing a new task. This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

**Parameters:**

*thread* Pointer to the Pooled Thread that is making this call

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2972).

#### 6.812.4.11 **virtual void decaf::util::concurrent::ThreadPool::queueTask** (Task *task*) throw ( lang::Exception ) [virtual]

**Queue** (p. 3149) a task to be completed by one of the Pooled Threads. tasks are serviced as soon as a PooledThread (p. 2968) is available to run it.

**Parameters:**

*task* object that derives from Runnable

**Exceptions:**

*ActiveMQException*

#### 6.812.4.12 **virtual void decaf::util::concurrent::ThreadPool::reserve** (std::size\_t *size*) [virtual]

Ensures that there is at least the specified number of Threads allocated to the pool. If the size is greater than the MAX number of threads in the pool, then only MAX threads are reserved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

**Parameters:**

*size* the number of threads to reserve.

#### 6.812.4.13 **virtual void decaf::util::concurrent::ThreadPool::setBlockSize** (std::size\_t *blockSize*) [virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

**Parameters:**

*blockSize* Max Thread Block Size

**6.812.4.14**    `virtual void decaf::util::concurrent::ThreadPool::setMaxThreads`  
                  (`std::size_t maxThreads`)    [virtual]

Sets the Max number of threads this pool can contain. if this value is smaller than the current size of the pool nothing is done.

**Parameters:**

*maxThreads* total number of threads that can be pooled

## 6.812.5    Field Documentation

**6.812.5.1**    `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3`    [static]

**6.812.5.2**    `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10`    [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPool.h`

## 6.813 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

#include <src/main/decaf/lang/Throwable.h> Inheritance diagram for decaf::lang::Throwable:

### Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0  
*Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.*
- virtual const std::exception \* **getCause** () const =0  
*Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 159) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \*cause)=0  
*Initializes the contained cause exception with the one given.*
- virtual void **setMark** (const char \*file, const int lineNumber)=0  
*Adds a file/line number to the stack trace.*
- virtual **Throwable** \* **clone** () const =0  
*Clones this exception.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **printStackTrace** () const =0  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const =0  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const =0  
*Gets the stack trace as one contiguous string.*

### 6.813.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1831) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.



**Throwable** (p. 3783) can wrap another **Throwable** (p. 3783) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 3783) instance and will be deleted when it is deleted or goes out of scope.

**Since:**

1.0

## 6.813.2 Constructor & Destructor Documentation

**6.813.2.1** `decaf::lang::Throwable::Throwable () throw () [inline]`

**6.813.2.2** `virtual decaf::lang::Throwable::~~Throwable () throw () [inline, virtual]`

## 6.813.3 Member Function Documentation

**6.813.3.1** `virtual Throwable* decaf::lang::Throwable::clone () const [pure virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

Copy of this **Exception** (p. 1831) object

Implemented	in	<b>activemq::exceptions::ActiveMQException</b>
(p. 358),	<b>activemq::exceptions::BrokerException</b>	(p. 867), <b>de-</b>
<b>caf::internal::net::ssl::openssl::OpenSSLSocketException</b>	(p. 2873),	<b>de-</b>
<b>caf::io::EOFException</b>	(p. 1827), <b>decaf::io::InterruptedIOException</b>	(p. 2129),
<b>decaf::io::IOException</b>	(p. 2144), <b>decaf::io::UnsupportedEncodingException</b>	(p. 3968), <b>de-</b>
(p. 3915), <b>decaf::io::UTFDataFormatException</b>	(p. 1834),	<b>de-</b>
<b>caf::lang::Exception</b>	(p. 1834), <b>decaf::lang::exceptions::ClassCastException</b>	(p. 1993),
(p. 1147), <b>decaf::lang::exceptions::IllegalArgumentException</b>	(p. 1996),	<b>de-</b>
<b>decaf::lang::exceptions::IllegalMonitorStateException</b>	(p. 2000),	<b>de-</b>
<b>caf::lang::exceptions::IllegalStateException</b>	(p. 2003),	<b>de-</b>
<b>caf::lang::exceptions::IndexOutOfBoundsException</b>	(p. 2010),	<b>de-</b>
<b>caf::lang::exceptions::InterruptedException</b>	(p. 2126),	<b>de-</b>
<b>caf::lang::exceptions::InvalidStateException</b>	(p. 2141),	<b>de-</b>
<b>caf::lang::exceptions::NoSuchElementException</b>	(p. 2828),	<b>de-</b>
<b>caf::lang::exceptions::NullPointerException</b>	(p. 2834),	<b>de-</b>
<b>caf::lang::exceptions::NumberFormatException</b>	(p. 2840),	<b>de-</b>
<b>decaf::lang::exceptions::RuntimeException</b>	(p. 3330),	<b>de-</b>
<b>caf::lang::exceptions::UnsupportedOperationException</b>	(p. 3918),	<b>de-</b>
<b>caf::net::BindException</b>	(p. 838), <b>decaf::net::ConnectException</b>	(p. 1261), <b>de-</b>
<b>caf::net::HttpRetryException</b>	(p. 1988), <b>decaf::net::MalformedURLException</b>	(p. 2458),
<b>decaf::net::NoRouteToHostException</b>	(p. 2822), <b>decaf::net::PortUnreachableException</b>	(p. 3522),
<b>decaf::net::ProtocolException</b>	(p. 3139), <b>decaf::net::SocketException</b>	(p. 3522),
<b>decaf::net::SocketTimeoutException</b>	(p. 3544), <b>decaf::net::UnknownHostException</b>	(p. 3909),
<b>decaf::net::UnknownServiceException</b>	(p. 3912),	<b>de-</b>
<b>caf::net::URISyntaxException</b>	(p. 3950), <b>decaf::nio::BufferOverflowException</b>	(p. 956),
<b>decaf::nio::BufferUnderflowException</b>	(p. 959),	<b>de-</b>
<b>caf::nio::InvalidMarkException</b>	(p. 2137), <b>decaf::nio::ReadOnlyBufferException</b>	

(p. 3171), **decaf::security::cert::CertificateEncodingException**  
 (p. 1091), **decaf::security::cert::CertificateException** (p. 1093), **de-**  
**caf::security::cert::CertificateExpiredException** (p. 1095), **de-**  
**caf::security::cert::CertificateNotYetValidException** (p. 1097),  
**decaf::security::cert::CertificateParsingException** (p. 1099),  
**decaf::security::GeneralSecurityException** (p. 1973), **de-**  
**caf::security::InvalidKeyException** (p. 2134), **decaf::security::KeyException**  
 (p. 2297), **decaf::security::KeyManagementException** (p. 2300),  
**decaf::security::NoSuchAlgorithmException** (p. 2825), **de-**  
**caf::security::NoSuchProviderException** (p. 2831), **decaf::security::SignatureException**  
 (p. 3499), **decaf::util::concurrent::BrokenBarrierException** (p. 862),  
**decaf::util::concurrent::CancellationException** (p. 1085), **de-**  
**caf::util::concurrent::ExecutionException** (p. 1868), **de-**  
**caf::util::concurrent::RejectedExecutionException** (p. 3191),  
**decaf::util::concurrent::TimeoutException** (p. 3789), **de-**  
**caf::util::zip::DataFormatException** (p. 1557), and **decaf::util::zip::ZipException**  
 (p. 4066).

**6.813.3.2 virtual const std::exception\* decaf::lang::Throwable::getCause () const**  
 [pure virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 159) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

**Returns:**

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1835).

**6.813.3.3 virtual std::string decaf::lang::Throwable::getMessage () const** [pure virtual]

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.

**Returns:**

string errors message

Implemented in **decaf::lang::Exception** (p. 1835).

**6.813.3.4 virtual std::vector< std::pair< std::string, int> >**  
**decaf::lang::Throwable::getStackTrace () const** [pure virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

**Returns:**

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1835).

**6.813.3.5 virtual std::string decaf::lang::Throwable::getStackTraceString () const** [pure virtual]

Gets the stack trace as one contiguous string.

**Returns:**

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1835).

**6.813.3.6 virtual void decaf::lang::Throwable::initCause (const std::exception \* *cause*)** [pure virtual]

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

**Parameters:**

*cause* The exception that was the cause of this one.

Implemented in **decaf::lang::Exception** (p. 1836).

**6.813.3.7 virtual void decaf::lang::Throwable::printStackTrace (std::ostream & *stream*) const** [pure virtual]

Prints the stack trace to the given output stream.

**Parameters:**

*stream* the target output stream.

Implemented in **decaf::lang::Exception** (p. 1836).

**6.813.3.8 virtual void decaf::lang::Throwable::printStackTrace () const** [pure virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1836).

**6.813.3.9 virtual void decaf::lang::Throwable::setMark (const char \* *file*, const int *lineNumber*)** [pure virtual]

Adds a file/line number to the stack trace.

**Parameters:**

*file* The name of the file calling this method (use `__FILE__`).

*lineNumber* The line number in the calling file (use `__LINE__`).

Implemented in **decaf::lang::Exception** (p. 1836).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Throwable.h`

## 6.814 decaf::util::concurrent::TimeoutException Class Reference

#include <src/main/decaf/util/concurrent/TimeoutException.h> Inheritance diagram for decaf::util::concurrent::TimeoutException:

### Public Member Functions

- **TimeoutException** () throw ()  
*Default Constructor.*
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **TimeoutException** (const **TimeoutException** &ex) throw ()  
*Copy Constructor.*
- **TimeoutException** (const std::exception \*cause) throw ()  
*Constructor.*
- **TimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **TimeoutException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **TimeoutException \* clone** () const  
*Clones this exception.*
- virtual ~**TimeoutException** () throw ()

### 6.814.1 Constructor & Destructor Documentation

#### 6.814.1.1 decaf::util::concurrent::TimeoutException::TimeoutException () throw () [inline]

Default Constructor.

#### 6.814.1.2 decaf::util::concurrent::TimeoutException::TimeoutException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.814.1.3 decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* The exception to copy from.

References decaf::lang::Exception::Exception().

**6.814.1.4 decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.814.1.5 decaf::util::concurrent::TimeoutException::TimeoutException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The string message to report

... list of primitives that are formatted into the message

**6.814.1.6 decaf::util::concurrent::TimeoutException::TimeoutException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The string message to report

... list of primitives that are formatted into the message

**6.814.1.7** `virtual decaf::util::concurrent::TimeoutException::~~TimeoutException ()  
throw () [inline, virtual]`

## **6.814.2 Member Function Documentation**

**6.814.2.1** `virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone () const  
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new **TimeoutException** (p.3787) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1834).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

## 6.815 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

### Public Member Functions

- **Timer** ()
- virtual **~Timer** ()
- void **cancel** ()  
*Terminates this timer, discarding any currently scheduled tasks.*
- std::size\_t **purge** ()  
*Removes all canceled tasks from this timer's task queue.*
- void **schedule** (**TimerTask** \*task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution after the specified delay.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution after the specified delay.*
- void **schedule** (**TimerTask** \*task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution at the specified time.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution at the specified time.*
- void **schedule** (**TimerTask** \*task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*

- void **schedule** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*

- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*

- void **scheduleAtFixedRate** (**TimerTask** \*task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*

- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*

- void **scheduleAtFixedRate** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*

- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*

### 6.815.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 3790) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 3790) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 3790) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the wait(long) method.



Since:

1.0

## 6.815.2 Constructor & Destructor Documentation

**6.815.2.1** `decaf::util::Timer::Timer ()`

**6.815.2.2** `virtual decaf::util::Timer::~~Timer ()` [virtual]

## 6.815.3 Member Function Documentation

**6.815.3.1** `void decaf::util::Timer::cancel ()`

Terminates this timer, discarding any currently scheduled tasks. Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the run method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

**6.815.3.2** `std::size_t decaf::util::Timer::purge ()`

Removes all canceled tasks from this timer's task queue. Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p. 3790) to destroy the **TimerTask** (p. 3801) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to  $n + c \log n$ , where  $n$  is the number of tasks in the queue and  $c$  is the number of canceled tasks.

This method can be called on a **Timer** (p. 3790) object that has no scheduled tasks without error.

**Returns:**

the number of tasks removed from the queue.

**6.815.3.3** `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, const Date & firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity),

subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters:

*task* - task to be scheduled.

*firstTime* - First time at which task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

***NullPointerException*** - if the **TimerTask** (p. 3801) value is Null.

***IllegalArgumentException*** - if `time.getTime()` is negative.

***IllegalStateException*** - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.4** `void decaf::util::Timer::schedule (TimerTask * task,  
const Date & firstTime, long long period) throw  
( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters:

*task* - task to be scheduled.

*firstTime* - First time at which task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if time.getTime() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.5** void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters:

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if delay is negative, or delay + System.currentTimeMillis() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.6** void decaf::util::Timer::schedule (TimerTask \* task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals separated by the specified

period.

The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters:

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

**NullPointerException** - if the **TimerTask** (p. 3801) value is Null.

**IllegalArgumentException** - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

**IllegalStateException** - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.7 void decaf::util::Timer::schedule (const decaf::lang::Pointer<
TimerTask > & task, const Date & time) throw
( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution at the specified time. If the time is in the past, the task is scheduled for immediate execution.

#### Parameters:

*task* - task to be scheduled.

*time* - time at which task is to be executed.

#### Exceptions:

**NullPointerException** - if the **TimerTask** (p. 3801) value is Null.

**IllegalArgumentException** - if `time.getTime()` is negative.

**IllegalStateException** - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.8** void decaf::util::Timer::schedule (TimerTask \* *task*, const Date & *time*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for execution at the specified time. If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.

**Parameters:**

*task* - task to be scheduled.

*time* - time at which task is to be executed.

**Exceptions:**

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if time.getTime() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.9** void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task*, long long *delay*) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for execution after the specified delay.

**Parameters:**

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

**Exceptions:**

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if delay is negative, or delay + System.currentTimeMillis() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, or timer was cancelled.

**6.815.3.10** `void decaf::util::Timer::schedule (TimerTask * task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for execution after the specified delay. The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.

**Parameters:**

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

**Exceptions:**

**NullPointerException** - if the **TimerTask** (p. 3801) value is Null.

**IllegalArgumentException** - if delay is negative, or delay + System.currentTimeMillis() is negative.

**IllegalStateException** - if task was already scheduled or cancelled, or timer was cancelled.

**6.815.3.11** `void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & task, const Date & firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

**Parameters:**

*task* - task to be scheduled.

*firstTime* - First time at which task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if time.getTime() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.12** void decaf::util::Timer::scheduleAtFixedRate (TimerTask  
\* task, const Date & firstTime, long long period)  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

#### Parameters:

*task* - task to be scheduled.

*firstTime* - First time at which task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if time.getTime() is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.13** `void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

#### Parameters:

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

*period* - time in milliseconds between successive task executions.

#### Exceptions:

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.14** `void decaf::util::Timer::scheduleAtFixedRate (TimerTask * task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3801) pointer is considered to be owned by the **Timer** (p. 3790) class once it has been scheduled, the **Timer** (p. 3790) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3790) itself is cancelled. A **TimerTask** (p. 3801) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3790) and the caller should ensure that the **TimerTask** (p. 3801) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3801) instance are planned.



In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

**Parameters:**

*task* - task to be scheduled.

*delay* - delay in milliseconds before task is to be executed.

*period* - time in milliseconds between successive task executions.

**Exceptions:**

*NullPointerException* - if the **TimerTask** (p. 3801) value is Null.

*IllegalArgumentException* - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

*IllegalStateException* - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Timer.h`

## 6.816 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3790).

`#include <src/main/decaf/util/TimerTask.h>`Inheritance diagram for decaf::util::TimerTask:

### Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()  
*Cancels this timer task.*
- long long **scheduledExecutionTime** () const  
*Returns the scheduled execution time of the most recent actual execution of this task.*

### Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

### Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

#### 6.816.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3790).

**Since:**

1.0

## 6.816.2 Constructor & Destructor Documentation

### 6.816.2.1 decaf::util::TimerTask::TimerTask ()

### 6.816.2.2 virtual decaf::util::TimerTask::~~TimerTask () [inline, virtual]

## 6.816.3 Member Function Documentation

### 6.816.3.1 bool decaf::util::TimerTask::cancel ()

Cancels this timer task. If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

#### Returns:

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

### 6.816.3.2 long long decaf::util::TimerTask::getWhen () const [protected]

### 6.816.3.3 bool decaf::util::TimerTask::isScheduled () const [protected]

### 6.816.3.4 long long decaf::util::TimerTask::scheduledExecutionTime () const

Returns the scheduled execution time of the most recent actual execution of this task. (If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 3325) { if( System::currentTimeMillis() - scheduledExecutionTime() (p. 3802) >= MAX_TARDINESS) return; // Too late; skip this execution. // Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

#### Returns:

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1667). The return value is undefined if the task has yet to commence its first execution.

**6.816.3.5** void decaf::util::TimerTask::setScheduledTime (long long *time*)  
[protected]

## **6.816.4 Friends And Related Function Documentation**

**6.816.4.1** friend class decaf::internal::util::TimerTaskHeap [friend]

**6.816.4.2** friend class Timer [friend]

**6.816.4.3** friend class TimerImpl [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**TimerTask.h**

## 6.817 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

### Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer**< **TimerTask** > **peek** ()

*Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.*

- bool **isEmpty** () const
- std::size\_t **size** () const
- void **insert** (const **Pointer**< **TimerTask** > &task)

*Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.*

- void **remove** (std::size\_t pos)

*Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.*

- void **reset** ()

*Clear all contents from the heap.*

- void **adjustMinimum** ()

*Resorts the heap starting at the top.*

- std::size\_t **deleteIfCancelled** ()

*Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.*

- std::size\_t **find** (const **Pointer**< **TimerTask** > &task) const

*Searches the heap for the specified TimerTask element and returns its position in the heap.*

### 6.817.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since:

1.0

## 6.817.2 Constructor & Destructor Documentation

**6.817.2.1** `decaf::internal::util::TimerTaskHeap::TimerTaskHeap ()`

**6.817.2.2** `virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ()`  
[virtual]

## 6.817.3 Member Function Documentation

**6.817.3.1** `void decaf::internal::util::TimerTaskHeap::adjustMinimum ()`

Resorts the heap starting at the top.

**6.817.3.2** `std::size_t decaf::internal::util::TimerTaskHeap::deleteIfCancelled ()`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

### Returns:

the number of task that were removed from the heap because they were cancelled.

**6.817.3.3** `std::size_t decaf::internal::util::TimerTaskHeap::find (const Pointer< TimerTask > & task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap. Returns the unsigned equivalent of -1 if the element is not found.

### Returns:

the position in the Heap where the Task is stored, or npos.

**6.817.3.4** `void decaf::internal::util::TimerTaskHeap::insert (const Pointer< TimerTask > & task)`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

### Parameters:

*task* The TimerTask to insert into the heap.

**6.817.3.5** `bool decaf::internal::util::TimerTaskHeap::isEmpty () const`

### Returns:

true if the heap is empty.

**6.817.3.6** `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ()`

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

**Returns:**

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

**6.817.3.7** `void decaf::internal::util::TimerTaskHeap::remove (std::size_t pos)`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

**Parameters:**

*pos* The position at which to remove the TimerTask and begin a resort of the heap.

**6.817.3.8** `void decaf::internal::util::TimerTaskHeap::reset ()`

Clear all contents from the heap.

**6.817.3.9** `std::size_t decaf::internal::util::TimerTaskHeap::size () const`**Returns:**

the size of the heap.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/TimerTaskHeap.h`

## 6.818 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 3807) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

#include <src/main/decaf/util/concurrent/TimeUnit.h> Inheritance diagram for decaf::util::concurrent::TimeUnit:

### Public Member Functions

- virtual **~TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const  
*Convert the given time duration in the given unit to this unit.*
- long long **toNanos** (long long duration) const  
*Equivalent to `NANOSECONDS.convert(duration, this)`.*
- long long **toMicros** (long long duration) const  
*Equivalent to `MICROSECONDS.convert(duration, this)`.*
- long long **toMillis** (long long duration) const  
*Equivalent to `MILLISECONDS.convert(duration, this)`.*
- long long **toSeconds** (long long duration) const  
*Equivalent to `SECONDS.convert(duration, this)`.*
- long long **toMinutes** (long long duration) const  
*Equivalent to `MINUTES.convert(duration, this)`.*
- long long **toHours** (long long duration) const  
*Equivalent to `HOURS.convert(duration, this)`.*
- long long **toDays** (long long duration) const  
*Equivalent to `DAYS.convert(duration, this)`.*
- void **timedWait** (**Synchronizable** \*obj, long long timeout) const throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed `Object.wait` using this time unit.*
- void **timedJoin** (decaf::lang::Thread \*thread, long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed `Thread.join` using this time unit.*
- void **sleep** (long long timeout) const throw ( decaf::lang::exceptions::InterruptedException )  
*Perform a `Thread.sleep` using this unit.*



- virtual std::string **toString** () const  
*Converts the **TimeUnit** (p. 3807) type to the Name of the **TimeUnit** (p. 3807).*
- virtual int **compareTo** (const **TimeUnit** &value) const  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **TimeUnit** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

## Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Returns the **TimeUnit** (p. 3807) constant of this type with the specified name.*

## Static Public Attributes

- static const **TimeUnit** NANoseconds  
*The Actual **TimeUnit** (p. 3807) enumerations.*
- static const **TimeUnit** MICROseconds
- static const **TimeUnit** MILLIseconds
- static const **TimeUnit** SECONDS
- static const **TimeUnit** MINUTES
- static const **TimeUnit** HOURS
- static const **TimeUnit** DAYS
- static const **TimeUnit** \*const **values** []  
*The An Array of **TimeUnit** (p. 3807) Instances.*

## Protected Member Functions

- **TimeUnit** (int index, const std::string &name)  
*Hidden Constructor, this class can not be instantiated directly.*

### 6.818.1 Detailed Description

A **TimeUnit** (p. 3807) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 3807) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is

defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 3807) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following **code** (p. 1183) will timeout in 50 milliseconds if the lock is not available:

**Lock** (p. 2375) lock = ...; if ( lock.tryLock( 50, **TimeUnit.MILLISECONDS** (p. 3815) ) ) ...

while this **code** (p. 1183) will timeout in 50 seconds:

**Lock** (p. 2375) lock = ...; if ( lock.tryLock( 50, **TimeUnit.SECONDS** (p. 3816) ) ) ...

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3807).

## 6.818.2 Constructor & Destructor Documentation

### 6.818.2.1 decaf::util::concurrent::TimeUnit::TimeUnit (int *index*, const std::string & *name*) [protected]

Hidden Constructor, this class can not be instantiated directly.

#### Parameters:

*index* - Index into the Time Unit set.

*name* - Name of the unit type being represented.

### 6.818.2.2 virtual decaf::util::concurrent::TimeUnit::~~TimeUnit () [inline, virtual]

## 6.818.3 Member Function Documentation

### 6.818.3.1 virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation  $\text{sgn}(\text{expression})$  designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure  $\text{sgn}(x.\text{compareTo}(y)) == -\text{sgn}(y.\text{compareTo}(x))$  for all  $x$  and  $y$ . (This implies that  $x.\text{compareTo}(y)$  must throw an exception iff  $y.\text{compareTo}(x)$  throws an exception.)

The implementor must also ensure that the relation is transitive:  $(x.\text{compareTo}(y) > 0 \ \&\& \ y.\text{compareTo}(z) > 0)$  implies  $x.\text{compareTo}(z) > 0$ .

Finally, the implementor must ensure that  $x.\text{compareTo}(y) == 0$  implies that  $\text{sgn}(x.\text{compareTo}(z)) == \text{sgn}(y.\text{compareTo}(z))$ , for all  $z$ .

It is strongly recommended, but not strictly required that  $(x.\text{compareTo}(y) == 0) == (x.\text{equals}(y))$ . Generally speaking, any class that implements the Comparable interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

**Parameters:**

*value* - the Object to be compared.

**Returns:**

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

**6.818.3.2 long long decaf::util::concurrent::TimeUnit::convert (long long *sourceDuration*, const TimeUnit & *sourceUnit*) const**

Convert the given time duration in the given unit to this unit. Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN\_VALUE if negative or Long.MAX\_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES (p. 3815))

**Parameters:**

*sourceDuration* - Duration value to convert.

*sourceUnit* - Unit type of the source duration.

**Returns:**

the converted duration in this unit, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.

**6.818.3.3 virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & *value*) const [virtual]****Returns:**

true if this value is considered equal to the passed value.

**6.818.3.4 virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & *value*) const [virtual]**

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.818.3.5** `virtual bool decaf::util::concurrent::TimeUnit::operator==(const  
TimeUnit & value) const` [virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.818.3.6** `void decaf::util::concurrent::TimeUnit::sleep (long long timeout) const  
throw ( decaf::lang::exceptions::InterruptedException )`

Perform a `Thread.sleep` using this unit. This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

**Parameters:**

*timeout* the minimum time to sleep

**See also:**

`Thread::sleep`

**6.818.3.7** `void decaf::util::concurrent::TimeUnit::timedJoin  
(decaf::lang::Thread * thread, long long timeout)  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException )`

Perform a timed `Thread.join` using this time unit. This is a convenience method that converts time arguments into the form required by the `Thread.join` method.

**Parameters:**

*thread* the thread to wait for

*timeout* the maximum time to wait

**Exceptions:**

*InterruptedException* if interrupted while waiting.

*NullPointerException* if the thread object is null.

**See also:**

`Thread::join( long long, long long )`

**6.818.3.8** void decaf::util::concurrent::TimeUnit::timedWait  
(Synchronizable \* *obj*, long long *timeout*) const  
throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException )

Perform a timed `Object.wait` using this time unit. This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking `poll` method (see `BlockingQueue.poll` (p. 847)) using:

```
Object poll( long long timeout, const TimeUnit& unit )  
    throw( InterruptedException ) {  
  
    while( empty ) {  
        unit.timedWait(this, timeout);  
        ...  
    }  
}
```

**Parameters:**

*obj* the object to wait on

*timeout* the maximum time to wait.

**Exceptions:**

*InterruptedException* if interrupted while waiting.

*NullPointerException* if the `Synchronizable` (p. 3700) object is null.

**See also:**

`Synchronizable::wait( long long, long long )`

**6.818.3.9** long long decaf::util::concurrent::TimeUnit::toDays (long long *duration*)  
const [inline]

Equivalent to `DAYS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration.

**See also:**

`convert` (p. 3810)

**6.818.3.10** `long long decaf::util::concurrent::TimeUnit::toHours (long long duration) const [inline]`

Equivalent to `HOURS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration.

**See also:**

`convert` (p. 3810)

**6.818.3.11** `long long decaf::util::concurrent::TimeUnit::toMicros (long long duration) const [inline]`

Equivalent to `MICROSECONDS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**See also:**

`convert` (p. 3810)

**6.818.3.12** `long long decaf::util::concurrent::TimeUnit::toMillis (long long duration) const [inline]`

Equivalent to `MILLISECONDS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**See also:**

`convert` (p. 3810)

**6.818.3.13** `long long decaf::util::concurrent::TimeUnit::toMinutes (long long duration) const [inline]`

Equivalent to `MINUTES.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration.

**See also:**

`convert` (p. 3810)

**6.818.3.14** `long long decaf::util::concurrent::TimeUnit::toNanos (long long duration) const [inline]`

Equivalent to `NANOSECONDS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**See also:**

`convert` (p. 3810)

**6.818.3.15** `long long decaf::util::concurrent::TimeUnit::toSeconds (long long duration) const [inline]`

Equivalent to `SECONDS.convert(duration, this)`.

**Parameters:**

*duration* the duration

**Returns:**

the converted duration.

**See also:**

`convert` (p. 3810)

**6.818.3.16** `virtual std::string decaf::util::concurrent::TimeUnit::toString () const`  
[virtual]

Converts the **TimeUnit** (p. 3807) type to the Name of the **TimeUnit** (p. 3807).

**Returns:**

String name of the **TimeUnit** (p. 3807)

**6.818.3.17** `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf`  
(const std::string & *name*) throw ( de-  
caf::lang::exceptions::IllegalArgumentException )  
[static]

Returns the **TimeUnit** (p. 3807) constant of this type with the specified name. The string must match exactly an identifier used to declare an **TimeUnit** (p. 3807) constant in this type. (Extraaneous whitespace characters are not permitted.)

**Parameters:**

*name* The Name of the **TimeUnit** (p. 3807) constant to be returned.

**Returns:**

A constant reference to the **TimeUnit** (p. 3807) Constant with the given name.

**Exceptions:**

*IllegalArgumentException* if this enum type has no constant with the specified name

## 6.818.4 Field Documentation

**6.818.4.1** `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

**6.818.4.2** `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

**6.818.4.3** `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`  
[static]

**6.818.4.4** `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`  
[static]

**6.818.4.5** `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES` [static]

**6.818.4.6** `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`  
[static]

The Actual **TimeUnit** (p. 3807) enumerations.



**6.818.4.7** `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS` [static]

**6.818.4.8** `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`  
[static]

The An Array of **TimeUnit** (p. 3807) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

## 6.819 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

#include <src/main/cms/Topic.h> Inheritance diagram for cms::Topic:

### Public Member Functions

- virtual `~Topic ()`
- virtual `std::string getTopicName () const =0 throw ( CMSEException )`  
*Gets the name of this topic.*

### 6.819.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p.3817) is a Publish / Subscribe type **Destination** (p.1723). All Messages sent to a **Topic** (p.3817) are broadcast to all Subscribers of that **Topic** (p.3817) unless the Subscriber defines a **Message** (p.2534) selector that filters out that **Message** (p.2534).

**Since:**

1.0

### 6.819.2 Constructor & Destructor Documentation

**6.819.2.1** virtual `cms::Topic::~~Topic ()` [inline, virtual]

### 6.819.3 Member Function Documentation

**6.819.3.1** virtual `std::string cms::Topic::getTopicName () const throw ( CMSEException )` [pure virtual]

Gets the name of this topic.

**Returns:**

The topic name.

**Exceptions:**

*CMSEException* (p. 1160) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQTopic` (p.694).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

## 6.820 activemq::state::Tracked Class Reference

#include <src/main/activemq/state/Tracked.h> Inheritance diagram for activemq::state::Tracked:

### Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

### 6.820.1 Constructor & Destructor Documentation

**6.820.1.1** **activemq::state::Tracked::Tracked** () [inline]

**6.820.1.2** **activemq::state::Tracked::Tracked** (const **Pointer**< **decaf::lang::Runnable** > & *runnable*)

**6.820.1.3** virtual **activemq::state::Tracked::~~Tracked** () [inline, virtual]

### 6.820.2 Member Function Documentation

**6.820.2.1** bool **activemq::state::Tracked::isWaitingForResponse** () const [inline]

**6.820.2.2** void **activemq::state::Tracked::onResponse** ()

The documentation for this class was generated from the following file:

- src/main/activemq/state/**Tracked.h**

## 6.821 activemq::commands::TransactionId Class Reference

#include <src/main/activemq/commands/TransactionId.h> Inheritance diagram for activemq::commands::TransactionId:

### Public Types

- typedef decaf::lang::PointerComparator< TransactionId > COMPARATOR

### Public Member Functions

- TransactionId ()
- TransactionId (const TransactionId &other)
- virtual ~TransactionId ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual TransactionId \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const DataStructure \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const DataStructure \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **compareTo** (const TransactionId &value) const
- virtual bool **equals** (const TransactionId &value) const
- virtual bool **operator==** (const TransactionId &value) const
- virtual bool **operator<** (const TransactionId &value) const
- TransactionId & **operator=** (const TransactionId &other)

### Static Public Attributes

- static const unsigned char ID\_TRANSACTIONID = 0

#### 6.821.1 Member Typedef Documentation

6.821.1.1 typedef decaf::lang::PointerComparator<TransactionId>  
activemq::commands::TransactionId::COMPARATOR

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2348), and **activemq::commands::XATransactionId** (p. 4032).

## 6.821.2 Constructor & Destructor Documentation

**6.821.2.1** `activemq::commands::TransactionId::TransactionId ()`

**6.821.2.2** `activemq::commands::TransactionId::TransactionId (const TransactionId & other)`

**6.821.2.3** `virtual activemq::commands::TransactionId::~~TransactionId ()` [virtual]

## 6.821.3 Member Function Documentation

**6.821.3.1** `virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2348), and `activemq::commands::XATransactionId` (p. 4032).

**6.821.3.2** `virtual int activemq::commands::TransactionId::compareTo (const TransactionId & value) const` [virtual]

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2348), and `activemq::commands::XATransactionId` (p. 4032).

**6.821.3.3** `virtual void activemq::commands::TransactionId::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2348), and `activemq::commands::XATransactionId` (p. 4032).

**6.821.3.4** `virtual bool activemq::commands::TransactionId::equals (const TransactionId & value) const` [virtual]

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2349), and `activemq::commands::XATransactionId` (p. 4033).

**6.821.3.5** `virtual bool activemq::commands::TransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2349), and `activemq::commands::XATransactionId` (p. 4033).

**6.821.3.6** `virtual unsigned char activemq::commands::TransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Implements `activemq::commands::DataStructure` (p. 1662).

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2349), and `activemq::commands::XATransactionId` (p. 4033).

**6.821.3.7** `virtual bool activemq::commands::TransactionId::operator< (const TransactionId & value) const` [virtual]

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2349), and `activemq::commands::XATransactionId` (p. 4034).

**6.821.3.8** `TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & other)`

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2349), and `activemq::commands::XATransactionId` (p. 4034).

**6.821.3.9** `virtual bool activemq::commands::TransactionId::operator== (const TransactionId & value) const` [virtual]

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2350), and `activemq::commands::XATransactionId` (p. 4034).

**6.821.3.10** `virtual std::string activemq::commands::TransactionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 833).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2350), and **activemq::commands::XATransactionId** (p. 4034).

**6.821.4 Field Documentation****6.821.4.1 const unsigned char activemq::commands::TransactionId::ID\_TRANSACTIONID = 0 [static]**

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionId.h`

## 6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3823).

#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.822.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3823).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.822.2 Constructor & Destructor Documentation

6.822.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.822.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.822.3 Member Function Documentation

6.822.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2364), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4057).

6.822.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2365), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4058).

**6.822.3.3** `virtual int activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2365), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4058).

**6.822.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2365), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4058).

**6.822.3.5**    **virtual void ac-**  
**tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal**  
**(OpenWireFormat \* *wireFormat*, commands::DataStructure**  
**\* *dataStructure*, decaf::io::DataInputStream \* *dataIn*,**  
**utils::BooleanStream \* *bs*) throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2366), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4059).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h`

## 6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3827).

#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.823.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3827).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.823.2 Constructor & Destructor Documentation

6.823.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.823.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.823.3 Member Function Documentation

6.823.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2372), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4049).

6.823.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2373), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4050).

**6.823.3.3** `virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2373), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4050).

**6.823.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference 3833

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2373), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4050).

**6.823.3.5** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2374), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4051).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h`

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3831).

#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.824.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3831).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.824.2 Constructor & Destructor Documentation

6.824.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.824.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.824.3 Member Function Documentation

6.824.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2356), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4041).

6.824.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2357), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4042).

**6.824.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2357), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4042).

**6.824.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference 3837

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2357), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4042).

**6.824.3.5** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

### Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2358), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4043).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h`

## 6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3835).

#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.825.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3835).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.825.2 Constructor & Destructor Documentation

6.825.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.825.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.825.3 Member Function Documentation

6.825.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2360), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4053).

6.825.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2361), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4054).

**6.825.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2361), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4054).

**6.825.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2361), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4054).

**6.825.3.5** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2362), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4055).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h`

## 6.826 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3839).

#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.826.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3839).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.826.2 Constructor & Destructor Documentation

6.826.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.826.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.826.3 Member Function Documentation

6.826.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2368), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4045).

6.826.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2369), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4046).

**6.826.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2369), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4046).

**6.826.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2369), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4046).

**6.826.3.5** `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2370), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4047).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h`

## 6.827 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3843).

#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.827.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3843).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.827.2 Constructor & Destructor Documentation

6.827.2.1 `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.827.2.2 `virtual activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

## 6.827.3 Member Function Documentation

6.827.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2352), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4037).

6.827.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

### Exceptions:

*IOException* if an error occurs during the `unmarshal`.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2353), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4038).

**6.827.3.3** `virtual int activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2353), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4038).

**6.827.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryReader that provides that data sink

*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1645).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2353), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4038).

**6.827.3.5** `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters:**

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1652).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2354), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4039).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h`

## 6.828 activemq::commands::TransactionInfo Class Reference

#include <src/main/activemq/commands/TransactionInfo.h> Inheritance diagram for activemq::commands::TransactionInfo:

### Public Member Functions

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **TransactionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_TRANSACTIONINFO** = 7



## Protected Attributes

- `Pointer< ConnectionId > connectionId`
- `Pointer< TransactionId > transactionId`
- unsigned char `type`

## 6.828.1 Constructor & Destructor Documentation

**6.828.1.1** `activemq::commands::TransactionInfo::TransactionInfo ()`

**6.828.1.2** `virtual activemq::commands::TransactionInfo::~~TransactionInfo ()`  
[virtual]

## 6.828.2 Member Function Documentation

**6.828.2.1** `virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure ()`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.828.2.2** `virtual void activemq::commands::TransactionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters:

*src* - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.828.2.3** `virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

- 6.828.2.4** `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ()`  
[virtual]
- 6.828.2.5** `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ()`  
const [virtual]
- 6.828.2.6** `virtual unsigned char activemq::commands::TransactionInfo::getDataSetType () const`  
[virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSet** (p. 1660) type copy.

Implements **activemq::commands::DataSet** (p. 1662).

- 6.828.2.7** `virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ()`  
[virtual]
- 6.828.2.8** `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ()`  
const [virtual]
- 6.828.2.9** `virtual unsigned char activemq::commands::TransactionInfo::getType ()`  
const [virtual]
- 6.828.2.10** `virtual bool activemq::commands::TransactionInfo::isTransactionInfo ()`  
const [inline, virtual]

**Returns:**

an answer of true to the **isTransactionInfo()** (p. 3849) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

- 6.828.2.11** `virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.828.2.12** `virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]
- 6.828.2.13** `virtual void activemq::commands::TransactionInfo::setType (unsigned char type)` [virtual]
- 6.828.2.14** `virtual std::string activemq::commands::TransactionInfo::toString ()`  
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

**6.828.2.15** `virtual Pointer<Command> activemq::commands::TransactionInfo::visit (activemq::state::CommandVisitor * visitor) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1198).

### 6.828.3 Field Documentation

**6.828.3.1** `Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId` [protected]

**6.828.3.2** `const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7` [static]

**6.828.3.3** `Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId` [protected]

**6.828.3.4** `unsigned char activemq::commands::TransactionInfo::type` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionInfo.h`

## 6.829 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3851).

#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.829.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3851).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.829.2 Constructor & Destructor Documentation

**6.829.2.1** `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

**6.829.2.2** `virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

## 6.829.3 Member Function Documentation

**6.829.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.829.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.829.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 787).

**6.829.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 788).

**6.829.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 789).

**6.829.3.6** virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 790).

**6.829.3.7** virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 791).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**

## 6.830 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3855).

#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.830.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3855).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.830.2 Constructor & Destructor Documentation

**6.830.2.1** `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller()` `[inline]`

**6.830.2.2** `virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` `[inline, virtual]`

## 6.830.3 Member Function Documentation

**6.830.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.830.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.830.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 780).

**6.830.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 781).

**6.830.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 782).

**6.830.3.6** virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 783).

**6.830.3.7** virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**

## 6.831 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3859).

#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.831.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3859).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.831.2 Constructor & Destructor Documentation

**6.831.2.1** `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller()` `[inline]`

**6.831.2.2** `virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` `[inline, virtual]`

## 6.831.3 Member Function Documentation

**6.831.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.831.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.831.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 766).

**6.831.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 767).

**6.831.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 768).

**6.831.3.6** virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 769).

**6.831.3.7** virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 770).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h

## 6.832 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3863).

#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.832.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3863).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.832.2 Constructor & Destructor Documentation

**6.832.2.1** `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::TransactionInfoMarshaller()` `[inline]`

**6.832.2.2** `virtual activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` `[inline, virtual]`

## 6.832.3 Member Function Documentation

**6.832.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.832.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.832.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 794).

**6.832.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 795).

**6.832.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 796).

**6.832.3.6** virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 797).

**6.832.3.7** virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 798).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**TransactionInfoMarshaller.h**

## 6.833 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3867).

#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.833.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3867).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.833.2 Constructor & Destructor Documentation

**6.833.2.1** `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller()` `[inline]`

**6.833.2.2** `virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` `[inline, virtual]`

## 6.833.3 Member Function Documentation

**6.833.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.833.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.833.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 773).

**6.833.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 774).

**6.833.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 775).

**6.833.3.6** virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 776).

**6.833.3.7** virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h

## 6.834 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3871).

#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.834.1 Detailed Description

Marshaling `code` (p.1183) for Open Wire Format for **TransactionInfoMarshaller** (p.3871).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.834.2 Constructor & Destructor Documentation

**6.834.2.1** `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller()` `[inline]`

**6.834.2.2** `virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` `[inline, virtual]`

## 6.834.3 Member Function Documentation

**6.834.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.834.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.834.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 801).

**6.834.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 802).

**6.834.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 803).

**6.834.3.6** virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 804).

**6.834.3.7** virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h

## 6.835 activemq::state::TransactionState Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

### Public Member Functions

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual **~TransactionState** ()
- **std::string toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)
- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const
- void **addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- **std::vector**< **Pointer**< **ProducerState** > > **getProducerStates** ()

## 6.835.1 Constructor & Destructor Documentation

6.835.1.1 `activemq::state::TransactionState::TransactionState (const Pointer< TransactionId > & id)`

6.835.1.2 `virtual activemq::state::TransactionState::~~TransactionState ()`  
[virtual]

## 6.835.2 Member Function Documentation

6.835.2.1 `void activemq::state::TransactionState::addCommand (const Pointer< Command > & operation)`

6.835.2.2 `void activemq::state::TransactionState::addProducerState (const Pointer< ProducerState > & producerState)`

6.835.2.3 `void activemq::state::TransactionState::checkShutdown () const`

6.835.2.4 `const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands () const`  
[inline]

6.835.2.5 `const Pointer<TransactionId>& activemq::state::TransactionState::getId () const` [inline]

6.835.2.6 `int activemq::state::TransactionState::getPreparedResult () const`  
[inline]

6.835.2.7 `std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ()`

6.835.2.8 `bool activemq::state::TransactionState::isPrepared () const` [inline]

6.835.2.9 `void activemq::state::TransactionState::setPrepared (bool prepared)`  
[inline]

6.835.2.10 `void activemq::state::TransactionState::setPreparedResult (int preparedResult)` [inline]

6.835.2.11 `void activemq::state::TransactionState::shutdown ()` [inline]

6.835.2.12 `std::string activemq::state::TransactionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

## 6.836 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared **internal** (p. 144) API for dual stacks and queues.

`#include <src/main/decaf/internal/util/concurrent/Transferer.h>`Inheritance diagram for `decaf::internal::util::concurrent::Transferer< E >`:

### 6.836.1 Detailed Description

`template<typename E> class decaf::internal::util::concurrent::Transferer< E >`

Shared **internal** (p. 144) API for dual stacks and queues.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/Transferer.h`

## 6.837 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

#include <src/main/decaf/internal/util/concurrent/TransferQueue.h> Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

### Public Member Functions

- **TransferQueue** ()

*Node class for **TransferQueue** (p. 3878).*

- virtual ~**TransferQueue** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a put.*

- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a take.*

### 6.837.1 Detailed Description

```
template<typename E> class decaf::internal::util::concurrent::TransferQueue< E >
```

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers. The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

### 6.837.2 Constructor & Destructor Documentation

#### 6.837.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue () [inline]

Node class for **TransferQueue** (p. 3878). Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

```
6.837.2.2  template<typename E > virtual
           decaf::internal::util::concurrent::TransferQueue< E
           >::~~TransferQueue () [inline, virtual]
```

### 6.837.3 Member Function Documentation

```
6.837.3.1  template<typename E > virtual E*
           decaf::internal::util::concurrent::TransferQueue< E
           >::transfer (bool timed, long long nanos) throw
           ( decaf::util::concurrent::TimeoutException, de-
           caf::lang::exceptions::InterruptedException ) [inline,
           virtual]
```

Performs a take.

#### Parameters:

*timed* if this operation should timeout  
*nanos* the timeout, in nanoseconds

#### Returns:

the item provided or received;

#### Exceptions:

***TimeoutException*** if the operation timed out waiting for the producer to offer an item.  
***InterruptedException*** if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3877).

```
6.837.3.2  template<typename E > virtual void
           decaf::internal::util::concurrent::TransferQueue< E
           >::transfer (E * e, bool timed, long long nanos)
           throw ( decaf::util::concurrent::TimeoutException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Performs a put.

#### Parameters:

*e* the item to be handed to a consumer;  
*timed* if this operation should timeout  
*nanos* the timeout, in nanoseconds

#### Exceptions:

***TimeoutException*** if the operation timed out waiting for the consumer to accept the item offered.  
***InterruptedException*** if the thread was interrupted while waiting for the consumer to accept the item offered.



Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3877).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferQueue.h`

## 6.838 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

#include <src/main/decaf/internal/util/concurrent/TransferStack.h> Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

### Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a put.*
- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a take.*

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

### 6.838.1 Constructor & Destructor Documentation

**6.838.1.1** template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack () [inline]

**6.838.1.2** template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack () [inline, virtual]

### 6.838.2 Member Function Documentation

**6.838.2.1** template<typename E > virtual E\* decaf::internal::util::concurrent::TransferStack< E >::transfer (bool *timed*, long long *nanos*) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]

Performs a take.

#### Parameters:

*timed* if this operation should timeout  
*nanos* the timeout, in nanoseconds

#### Returns:

the item provided or received;

**Exceptions:**

***TimeoutException*** if the operation timed out waiting for the producer to offer an item.

***InterruptedException*** if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3877).

```
6.838.2.2  template<typename E > virtual void
           decaf::internal::util::concurrent::TransferStack< E
           >::transfer (E * e,  bool timed,  long long nanos)
           throw ( decaf::util::concurrent::TimeoutException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Performs a put.

**Parameters:**

*e* the item to be handed to a consumer;

*timed* if this operation should timeout

*nanos* the timeout, in nanoseconds

**Exceptions:**

***TimeoutException*** if the operation timed out waiting for the consumer to accept the item offered.

***InterruptedException*** if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3877).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferStack.h**

## 6.839 activemq::transport::Transport Class Reference

Interface for a **transport** (p. 97) layer for command objects.

#include <src/main/activemq/transport/Transport.h> Inheritance diagram for activemq::transport::Transport:

### Public Member Functions

- virtual **~Transport** ()
- virtual void **start** ()=0 throw ( decaf::io::IOException )  
*Starts the **Transport** (p. 3883), the send methods of a **Transport** (p. 3883) will throw an exception if used before the **Transport** (p. 3883) is started.*
- virtual void **stop** ()=0 throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3883).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0  
*Sets the **WireFormat** instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)=0  
*Sets the observer of asynchronous events from this **transport** (p. 97).*
- virtual **TransportListener** \* **getTransportListener** () const =0  
*Gets the observer of asynchronous events from this **transport** (p. 97).*
- virtual **Transport** \* **narrow** (const std::type\_info &typeId)=0  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0  
*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0

*Is the **Transport** (p. 3883) Connected to its Broker.*

- virtual bool **isClosed** () const =0

*Has the **Transport** (p. 3883) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw ( decaf::io::IOException )  
*reconnect to another location*

### 6.839.1 Detailed Description

Interface for a **transport** (p. 97) layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3883) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3883) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

### 6.839.2 Constructor & Destructor Documentation

**6.839.2.1** virtual **activemq::transport::Transport::~~Transport** () [inline, virtual]

### 6.839.3 Member Function Documentation

**6.839.3.1** virtual std::string **activemq::transport::Transport::getRemoteAddress** ()  
const [pure virtual]

#### Returns:

the remote address for this connection

Implemented in **activemq::transport::failover::FailoverTransport** (p.1877), **activemq::transport::IOTransport** (p.2147), **activemq::transport::mock::MockTransport** (p.2771), and **activemq::transport::TransportFilter** (p. 3894).

**6.839.3.2** virtual TransportListener\* **activemq::transport::Transport::getTransportListener** ()  
const [pure virtual]

Gets the observer of asynchronous events from this **transport** (p. 97).

#### Returns:

the listener of **transport** (p. 97) events.

Implemented in **activemq::transport::failover::FailoverTransport** (p.1877), **activemq::transport::IOTransport** (p.2147), **activemq::transport::mock::MockTransport** (p.2772), and **activemq::transport::TransportFilter** (p. 3894).

### 6.839.3.3 virtual bool activemq::transport::Transport::isClosed () const [pure virtual]

Has the **Transport** (p. 3883) been shutdown and no longer usable.

#### Returns:

true if the **Transport** (p. 3883)

Implemented in **activemq::transport::failover::FailoverTransport** (p.1878), **activemq::transport::IOTransport** (p.2147), **activemq::transport::mock::MockTransport** (p.2772), **activemq::transport::tcp::TcpTransport** (p.3754), and **activemq::transport::TransportFilter** (p.3894).

### 6.839.3.4 virtual bool activemq::transport::Transport::isConnected () const [pure virtual]

Is the **Transport** (p. 3883) Connected to its Broker.

#### Returns:

true if a connection has been made.

Implemented in **activemq::transport::failover::FailoverTransport** (p.1878), **activemq::transport::IOTransport** (p.2148), **activemq::transport::mock::MockTransport** (p.2772), **activemq::transport::tcp::TcpTransport** (p.3754), and **activemq::transport::TransportFilter** (p.3894).

### 6.839.3.5 virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns:

true if the **Transport** (p. 3883) is fault tolerant.

Implemented in **activemq::transport::failover::FailoverTransport** (p.1878), **activemq::transport::IOTransport** (p.2148), **activemq::transport::mock::MockTransport** (p.2773), **activemq::transport::tcp::TcpTransport** (p.3755), and **activemq::transport::TransportFilter** (p.3895).

### 6.839.3.6 virtual Transport\* activemq::transport::Transport::narrow (const std::type\_info & *typeId*) [pure virtual]

Narrows down a Chain of Transports to a specific **Transport** (p.3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.

#### Parameters:

*typeId* - The type\_info of the Object we are searching for.

#### Returns:

the requested Object. or NULL if its not in this chain.

Implemented in `activemq::transport::failover::FailoverTransport` (p.1879), `activemq::transport::IOTransport` (p.2148), `activemq::transport::mock::MockTransport` (p.2773), and `activemq::transport::TransportFilter` (p.3895).

Referenced by `activemq::transport::failover::FailoverTransport::narrow()`.

**6.839.3.7** `virtual void activemq::transport::Transport::oneway (const Pointer< Command > & command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Sends a one-way command. Does not wait for any response from the broker.

#### Parameters:

*command* the command to be sent.

#### Exceptions:

*IOException* if an exception occurs during writing of the command.

*UnsupportedOperationException* if this method is not implemented by this `transport` (p.97).

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p.3292), `activemq::transport::failover::FailoverTransport` (p.1880), `activemq::transport::inactivity::InactivityMonitor` (p.2006), `activemq::transport::IOTransport` (p.2148), `activemq::transport::logging::LoggingTransport` (p.2404), `activemq::transport::mock::MockTransport` (p.2773), `activemq::transport::TransportFilter` (p.3895), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p.2902).

**6.839.3.8** `virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & uri) throw ( decaf::io::IOException ) [pure virtual]`

reconnect to another location

#### Parameters:

*uri*

#### Exceptions:

*IOException* on failure of if not supported

Implemented in `activemq::transport::failover::FailoverTransport` (p.1880), and `activemq::transport::TransportFilter` (p.3896).

**6.839.3.9** `virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* - The command to be sent.  
*timeout* - The time to wait for this response.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.  
*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3293), **activemq::transport::failover::FailoverTransport** (p. 1881), **activemq::transport::IOTransport** (p. 2149), **activemq::transport::logging::LoggingTransport** (p. 2404), **activemq::transport::mock::MockTransport** (p. 2774), **activemq::transport::TransportFilter** (p. 3896), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2903).

```
6.839.3.10 virtual Pointer<Response> activemq::transport::Transport::request
            (const Pointer< Command > & command)
            throw ( decaf::io::IOException, de-
                    caf::lang::exceptions::UnsupportedOperationException )
            [pure virtual]
```

Sends the given command to the broker and then waits for the response.

**Parameters:**

*command* the command to be sent.

**Returns:**

the response from the broker.

**Exceptions:**

*IOException* if an exception occurs during the read of the command.  
*UnsupportedOperationException* if this method is not implemented by this **transport** (p. 97).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3293), **activemq::transport::failover::FailoverTransport** (p. 1881), **activemq::transport::IOTransport** (p. 2149), **activemq::transport::logging::LoggingTransport** (p. 2405), **activemq::transport::mock::MockTransport** (p. 2775), **activemq::transport::TransportFilter** (p. 3897), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2903).

```
6.839.3.11 virtual void activemq::transport::Transport::setTransportListener
            (TransportListener * listener) [pure virtual]
```

Sets the observer of asynchronous events from this **transport** (p. 97).



**Parameters:**

*listener* the listener of **transport** (p. 97) events.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1883), **activemq::transport::IOTransport** (p. 2150), **activemq::transport::mock::MockTransport** (p. 2777), and **activemq::transport::TransportFilter** (p. 3897).

### 6.839.3.12 virtual void activemq::transport::Transport::setWireFormat (const Pointer< wireformat::WireFormat > & *wireFormat*) [pure virtual]

Sets the WireFormat instance to use.

**Parameters:**

*wireFormat* The WireFormat the object used to encode / decode **commands** (p. 87).

Implemented in **activemq::transport::IOTransport** (p. 2150), and **activemq::transport::TransportFilter** (p. 3897).

### 6.839.3.13 virtual void activemq::transport::Transport::start () throw ( decaf::io::IOException ) [pure virtual]

Starts the **Transport** (p. 3883), the send methods of a **Transport** (p. 3883) will throw an exception if used before the **Transport** (p. 3883) is started.

**Exceptions:**

*IOException* if and error occurs while starting the **Transport** (p. 3883).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3294), **activemq::transport::failover::FailoverTransport** (p. 1884), **activemq::transport::IOTransport** (p. 2150), **activemq::transport::mock::MockTransport** (p. 2777), **activemq::transport::TransportFilter** (p. 3898), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2904).

### 6.839.3.14 virtual void activemq::transport::Transport::stop () throw ( decaf::io::IOException ) [pure virtual]

Stops the **Transport** (p. 3883).

**Exceptions:**

*IOException* if an error occurs while stopping the **transport** (p. 97).

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1884), **activemq::transport::IOTransport** (p. 2151), **activemq::transport::mock::MockTransport** (p. 2777), and **activemq::transport::TransportFilter** (p. 3898).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**Transport.h**

## 6.840 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

#include <src/main/activemq/transport/TransportFactory.h> Inheritance diagram for activemq::transport::TransportFactory:

### Public Member Functions

- virtual **~TransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)=0 throw ( exceptions::ActiveMQException )  
*Creates a fully configured **Transport** (p. 3883) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)=0 throw ( exceptions::ActiveMQException )  
*Creates a slimed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.*

### 6.840.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters. The factory should be able to create either a completely configured **Transport** (p. 3883) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since:

3.0

### 6.840.2 Constructor & Destructor Documentation

- 6.840.2.1 **virtual activemq::transport::TransportFactory::~~TransportFactory** ()  
 [inline, virtual]

### 6.840.3 Member Function Documentation

- 6.840.3.1 **virtual Pointer<Transport> activemq::transport::TransportFactory::create** (const **decaf::net::URI** & *location*) throw ( exceptions::ActiveMQException )  
 [pure virtual]

Creates a fully configured **Transport** (p. 3883) instance which could be a chain of filters and transports.

Parameters:

*location* - URI location to connect to plus any properties to assign.

**Exceptions:**

*ActiveMQException* if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1886), **activemq::transport::mock::MockTransportFactory** (p. 2780), and **activemq::transport::tcp::TcpTransportFactory** (p. 3757).

**6.840.3.2** `virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & location) throw ( exceptions::ActiveMQException )`  
[pure virtual]

Creates a slimmed down **Transport** (p. 3883) instance which can be used in composite **transport** (p. 97) instances.

**Parameters:**

*location* - URI location to connect to plus any properties to assign.

**Exceptions:**

*ActiveMQException* if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1886), **activemq::transport::mock::MockTransportFactory** (p. 2780), and **activemq::transport::tcp::TcpTransportFactory** (p. 3757).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFactory.h`

## 6.841 activemq::transport::TransportFilter Class Reference

A filter on the **transport** (p. 97) layer.

#include <src/main/activemq/transport/TransportFilter.h> Inheritance diagram for activemq::transport::TransportFilter:

### Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)  
*Constructor.*
- virtual ~**TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **transportInterrupted** ()  
*The **transport** (p. 97) has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The **transport** (p. 97) has resumed after an interruption.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )  
*Not supported by this class - throws an exception.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )  
*Not supported by this class - throws an exception.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)  
*Sets the **WireFormat** instance to use.*
- virtual void **start** () throw ( **decaf::io::IOException** )

Starts this *transport* (p. 97) object and creates the thread for polling on the input stream for *commands* (p. 87).

- virtual void **stop** () throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3883).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual **Transport \* narrow** (const std::type\_info &typeId)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level *transport* (p. 97) to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3883) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3883) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri) throw ( decaf::io::IOException )  
*reconnect to another location*

## Protected Member Functions

- void **fire** (const decaf::lang::Exception &ex)  
*Notify the listener of the thrown Exception.*
- void **fire** (const Pointer< Command > &command)  
*Notify the listener of the new incoming Command.*

## Protected Attributes

- Pointer< Transport > **next**  
*The *transport* (p. 97) that this filter wraps around.*
- TransportListener \* **listener**  
*Listener of this *transport* (p. 97).*

### 6.841.1 Detailed Description

A filter on the **transport** (p. 97) layer. **Transport** (p. 3883) filters implement the **Transport** (p. 3883) interface and optionally delegate calls to another **Transport** (p. 3883) object.

**Since:**

1.0

### 6.841.2 Constructor & Destructor Documentation

#### 6.841.2.1 `activemq::transport::TransportFilter::TransportFilter (const Pointer< Transport > & next)`

Constructor.

**Parameters:**

*next* - the next **Transport** (p. 3883) in the chain

#### 6.841.2.2 `virtual activemq::transport::TransportFilter::~~TransportFilter ()` [inline, virtual]

### 6.841.3 Member Function Documentation

#### 6.841.3.1 `virtual void activemq::transport::TransportFilter::close () throw ( decaf::io::IOException )` [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions:**

*IOException* if an error occurs while closing the **Transport** (p. 3883).

Implements `decaf::io::Closeable` (p. 1149).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3292), `activemq::transport::inactivity::InactivityMonitor` (p. 2005), `activemq::transport::tcp::TcpTransport` (p. 3753), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2902).

#### 6.841.3.2 `void activemq::transport::TransportFilter::fire (const Pointer< Command > & command)` [protected]

Notify the listener of the new incoming Command.

**Parameters:**

*command* - the command to send to the listener

**6.841.3.3** `void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & ex)` [protected]

Notify the listener of the thrown Exception.

**Parameters:**

*ex* - the exception to send to listeners

**6.841.3.4** `virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const` [inline, virtual]

**Returns:**

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3884).

**6.841.3.5** `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous exceptions (p. 92) from this `transport` (p. 97).

**Returns:**

The listener of `transport` (p. 97) events.

Implements `activemq::transport::Transport` (p. 3884).

**6.841.3.6** `virtual bool activemq::transport::TransportFilter::isClosed () const` [inline, virtual]

Has the `Transport` (p. 3883) been shutdown and no longer usable.

**Returns:**

true if the `Transport` (p. 3883)

Implements `activemq::transport::Transport` (p. 3885).

Reimplemented in `activemq::transport::tcp::TcpTransport` (p. 3754).

**6.841.3.7** `virtual bool activemq::transport::TransportFilter::isConnected () const` [inline, virtual]

Is the `Transport` (p. 3883) Connected to its Broker.

**Returns:**

true if a connection has been made.

Implements `activemq::transport::Transport` (p. 3885).

Reimplemented in `activemq::transport::tcp::TcpTransport` (p. 3754).

### 6.841.3.8 virtual bool activemq::transport::TransportFilter::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 3883) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns:

true if the **Transport** (p. 3883) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3885).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3755).

### 6.841.3.9 virtual Transport\* activemq::transport::TransportFilter::narrow (const std::type\_info & *typeId*) [virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3883) to allow a higher level **transport** (p. 97) to skip intermediate Transports in certain circumstances.

#### Parameters:

*typeId* - The type\_info of the Object we are searching for.

#### Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3885).

### 6.841.3.10 virtual void activemq::transport::TransportFilter::onCommand (const Pointer< Command > & *command*) [virtual]

Event handler for the receipt of a command.

#### Parameters:

*command* - the received command object.

Implements **activemq::transport::TransportListener** (p. 3900).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3292), **activemq::transport::inactivity::InactivityMonitor** (p. 2005), **activemq::transport::logging::LoggingTransport** (p. 2404), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2902).

### 6.841.3.11 virtual void activemq::transport::TransportFilter::oneway (const Pointer< Command > & *command*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]

Sends a one-way command. Does not wait for any response from the broker.

#### Parameters:

*command* the command to be sent.



**Exceptions:**

***IOException*** if an exception occurs during writing of the command.

***UnsupportedOperationException*** if this method is not implemented by this **transport** (p. 97).

Implements **activemq::transport::Transport** (p. 3886).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3292), **activemq::transport::inactivity::InactivityMonitor** (p. 2006), **activemq::transport::logging::LoggingTransport** (p. 2404), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2902).

**6.841.3.12** **virtual void activemq::transport::TransportFilter::onException (const decaf::lang::Exception & *ex*)** [virtual]

Event handler for an exception from a command **transport** (p. 97).

**Parameters:**

*ex* The exception to handle.

Implements **activemq::transport::TransportListener** (p. 3901).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 2006).

**6.841.3.13** **virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & *uri*) throw ( decaf::io::IOException )** [virtual]

reconnect to another location

**Parameters:**

*uri*

**Exceptions:**

***IOException*** on failure of if not supported

Implements **activemq::transport::Transport** (p. 3886).

**6.841.3.14** **virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )** [inline, virtual]

Not supported by this class - throws an exception.

**Parameters:**

*command* - The command that is sent as a request

*timeout* - The the time to wait for a response.

**Exceptions:***IOException**UnsupportedOperationException.*Implements **activemq::transport::Transport** (p. 3886).Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3293), **activemq::transport::logging::LoggingTransport** (p. 2404), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2903).

**6.841.3.15** **virtual** **Pointer<Response>** **activemq::transport::TransportFilter::request** (**const** **Pointer< Command >** *& command*) **throw** (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**) [**inline**, **virtual**]

Not supported by this class - throws an exception.

**Parameters:***command* the command that is sent as a request**Exceptions:***IOException**UnsupportedOperationException.*Implements **activemq::transport::Transport** (p. 3887).Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3293), **activemq::transport::logging::LoggingTransport** (p. 2405), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2903).

**6.841.3.16** **virtual void** **activemq::transport::TransportFilter::setTransportListener** (**TransportListener \*** *listener*) [**inline**, **virtual**]

Sets the observer of asynchronous **exceptions** (p. 92) from this **transport** (p. 97).**Parameters:***listener* the listener of **transport** (p. 97) events.Implements **activemq::transport::Transport** (p. 3887).

**6.841.3.17** **virtual void** **activemq::transport::TransportFilter::setWireFormat** (**const** **Pointer< wireformat::WireFormat >** *& wireFormat*) [**inline**, **virtual**]

Sets the WireFormat instance to use.

**Parameters:***wireFormat* The WireFormat the object used to encode / decode **commands** (p. 87).Implements **activemq::transport::Transport** (p. 3888).

**6.841.3.18** `virtual void activemq::transport::TransportFilter::start () throw (decaf::io::IOException) [virtual]`

Starts this **transport** (p. 97) object and creates the thread for polling on the input stream for **commands** (p. 87). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions:**

*IOException* if an error occurs or if this **transport** (p. 97) has already been closed.

Implements **activemq::transport::Transport** (p. 3888).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3294), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2904).

**6.841.3.19** `virtual void activemq::transport::TransportFilter::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3883).

**Exceptions:**

*IOException* if an error occurs while stopping the **Transport** (p. 3883).

Implements **activemq::transport::Transport** (p. 3888).

**6.841.3.20** `virtual void activemq::transport::TransportFilter::transportInterrupted () [virtual]`

The **transport** (p. 97) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3901).

**6.841.3.21** `virtual void activemq::transport::TransportFilter::transportResumed () [virtual]`

The **transport** (p. 97) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3901).

## 6.841.4 Field Documentation

**6.841.4.1** `TransportListener* activemq::transport::TransportFilter::listener [protected]`

Listener of this **transport** (p. 97).

**6.841.4.2** `Pointer<Transport> activemq::transport::TransportFilter::next [protected]`

The **transport** (p. 97) that this filter wraps around.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFilter.h`

## 6.842 activemq::transport::TransportListener Class Reference

A listener of asynchronous **exceptions** (p. 92) from a command **transport** (p. 97) object.

#include <src/main/activemq/transport/TransportListener.h> Inheritance diagram for activemq::transport::TransportListener:

### Public Member Functions

- virtual **~TransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)=0  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)=0  
*Event handler for an exception from a command **transport** (p. 97).*
- virtual void **transportInterrupted** ()=0  
*The **transport** (p. 97) has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()=0  
*The **transport** (p. 97) has resumed after an interruption.*

### 6.842.1 Detailed Description

A listener of asynchronous **exceptions** (p. 92) from a command **transport** (p. 97) object.

### 6.842.2 Constructor & Destructor Documentation

- 6.842.2.1 **virtual activemq::transport::TransportListener::~~TransportListener** ()  
[inline, virtual]

### 6.842.3 Member Function Documentation

- 6.842.3.1 **virtual void activemq::transport::TransportListener::onCommand** (const **Pointer**< **Command** > & *command*) [pure virtual]

Event handler for the receipt of a command. The **transport** (p. 97) passes off all received **commands** (p. 87) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3883) deletes the command upon receipt.

#### Parameters:

*command* the received command object.

Implemented in **activemq::core::ActiveMQConnection** (p. 285), **ac-**  
**tivemq::transport::correlator::ResponseCorrelator** (p. 3292), **ac-**  
**tivemq::transport::failover::FailoverTransportListener** (p. 1889), **ac-**  
**tivemq::transport::inactivity::InactivityMonitor** (p. 2005), **ac-**  
**tivemq::transport::logging::LoggingTransport** (p. 2404), **ac-**  
**tivemq::transport::mock::InternalCommandListener** (p. 2122),  
**activemq::transport::TransportFilter** (p. 3895), and **ac-**  
**tivemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2902).

#### 6.842.3.2 virtual void activemq::transport::TransportListener::onException (const decaf::lang::Exception & *ex*) [pure virtual]

Event handler for an exception from a command **transport** (p. 97).

##### Parameters:

*ex* The exception being propagated to this listener to handle.

Implemented in **activemq::core::ActiveMQConnection** (p. 286),  
**activemq::transport::failover::BackupTransport** (p. 753), **ac-**  
**tivemq::transport::failover::FailoverTransportListener** (p. 1889), **ac-**  
**tivemq::transport::inactivity::InactivityMonitor** (p. 2006), and **ac-**  
**tivemq::transport::TransportFilter** (p. 3896).

#### 6.842.3.3 virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]

The **transport** (p. 97) has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 291),  
**activemq::transport::DefaultTransportListener** (p. 1705), **ac-**  
**tivemq::transport::failover::FailoverTransportListener** (p. 1889), and **ac-**  
**tivemq::transport::TransportFilter** (p. 3898).

#### 6.842.3.4 virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]

The **transport** (p. 97) has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 291),  
**activemq::transport::DefaultTransportListener** (p. 1705), **ac-**  
**tivemq::transport::failover::FailoverTransportListener** (p. 1889), and **ac-**  
**tivemq::transport::TransportFilter** (p. 3898).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

## 6.843 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3883) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

### Public Member Functions

- virtual **~TransportRegistry** ()
- **TransportFactory** \* **findFactory** (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets a Registered **TransportFactory** (p. 3889) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )

*Registers a new **TransportFactory** (p. 3889) with this Registry.*

- void **unregisterFactory** (const std::string &name)

*Unregisters the Factory with the given name and deletes that instance of the Factory.*

- std::vector< std::string > **getTransportNames** () const

*Retrieves a list of the names of all the Registered Transport's in this Registry.*

### Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 3902).*

#### 6.843.1 Detailed Description

Registry of all **Transport** (p. 3883) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since:

3.0

## 6.843.2 Constructor & Destructor Documentation

**6.843.2.1** `virtual activemq::transport::TransportRegistry::~~TransportRegistry ()`  
[virtual]

## 6.843.3 Member Function Documentation

**6.843.3.1** `TransportFactory* activemq::transport::TransportRegistry::findFactory`  
`(const std::string & name) const throw ( de-`  
`caf::lang::exceptions::NoSuchElementException )`

Gets a Registered **TransportFactory** (p. 3889) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

### Parameters:

*name* The name of the Factory to find in the Registry.

### Returns:

the Factory registered under the given name.

### Exceptions:

*NoSuchElementException* if no factory is registered with that name.

**6.843.3.2** `static TransportRegistry& ac-`  
`tivemq::transport::TransportRegistry::getInstance ()`  
[static]

Gets the single instance of the **TransportRegistry** (p. 3902).

### Returns:

reference to the single instance of this Registry

**6.843.3.3** `std::vector<std::string> ac-`  
`tivemq::transport::TransportRegistry::getTransportNames`  
`() const`

Retrieves a list of the names of all the Registered Transport's in this Registry.

### Returns:

std vector of strings with all the **Transport** (p. 3883) names registered.

**6.843.3.4** `void activemq::transport::TransportRegistry::registerFactory`  
`(const std::string & name, TransportFactory * factory)`  
`throw ( decaf::lang::exceptions::IllegalArgumentException,`  
`decaf::lang::exceptions::NullPointerException )`

Registers a new **TransportFactory** (p. 3889) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.



**Parameters:**

- name* The name of the new Factory to register.  
*factory* The new Factory to add to the Registry.

**Exceptions:**

- IllegalArgumentException* if name is the empty string.  
*NullPointerException* if the Factory is Null.

**6.843.3.5 void activemq::transport::TransportRegistry::unregisterFactory (const std::string & name)**

Unregisters the Factory with the given name and deletes that instance of the Factory.

**Parameters:**

- name* Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

## 6.844 tree\_desc\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- `ct_data * dyn_tree`
- `int max_code`
- `static_tree_desc * stat_desc`

### 6.844.1 Field Documentation

**6.844.1.1** `ct_data* tree_desc_s::dyn_tree`

**6.844.1.2** `int tree_desc_s::max_code`

**6.844.1.3** `static_tree_desc* tree_desc_s::stat_desc`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/deflate.h`

## 6.845 decaf::lang::Thread::UncaughtExceptionHandler Class Reference

Interface for handlers invoked when a **Thread** (p. 3765) abruptly terminates due to an uncaught exception.

```
#include <src/main/decaf/lang/Thread.h>
```

### Public Member Functions

- virtual **~UncaughtExceptionHandler** ()
- virtual void **uncaughtException** (const **Thread** \*thread, const **Throwable** &error)=0  
throw ()

*Method invoked when the given thread terminates due to the given uncaught exception.*

### 6.845.1 Detailed Description

Interface for handlers invoked when a **Thread** (p. 3765) abruptly terminates due to an uncaught exception.

### 6.845.2 Constructor & Destructor Documentation

- 6.845.2.1 virtual  
decaf::lang::Thread::UncaughtExceptionHandler::~UncaughtExceptionHandler  
() [inline, virtual]

### 6.845.3 Member Function Documentation

- 6.845.3.1 virtual void de-  
caf::lang::Thread::UncaughtExceptionHandler::uncaughtException (const  
**Thread** \* *thread*, const **Throwable** & *error*) throw () [pure virtual]

Method invoked when the given thread terminates due to the given uncaught exception. This method is defined to indicate that it will not throw an exception, throwing an exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

## 6.846 decaf::net::UnknownHostException Class Reference

#include <src/main/decaf/net/UnknownHostException.h> Inheritance diagram for decaf::net::UnknownHostException:

### Public Member Functions

- **UnknownHostException** () throw ()  
*Default Constructor.*
- **UnknownHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()  
*Copy Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownHostException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnknownHostException** \* clone () const  
*Clones this exception.*
- virtual ~**UnknownHostException** () throw ()

### 6.846.1 Constructor & Destructor Documentation

#### 6.846.1.1 decaf::net::UnknownHostException::UnknownHostException () throw () [inline]

Default Constructor.

#### 6.846.1.2 decaf::net::UnknownHostException::UnknownHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

**ex** An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.846.1.3 decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

**6.846.1.4 decaf::net::UnknownHostException::UnknownHostException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.846.1.5 decaf::net::UnknownHostException::UnknownHostException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.846.1.6 decaf::net::UnknownHostException::UnknownHostException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.846.1.7** `virtual decaf::net::UnknownHostException::~~UnknownHostException ()  
throw () [inline, virtual]`

## **6.846.2 Member Function Documentation**

**6.846.2.1** `virtual UnknownHostException* decaf::net::UnknownHostException::clone () const [inline,  
virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownHostException.h`

## 6.847 decaf::net::UnknownServiceException Class Reference

#include <src/main/decaf/net/UnknownServiceException.h> Inheritance diagram for decaf::net::UnknownServiceException:

### Public Member Functions

- **UnknownServiceException** () throw ()  
*Default Constructor.*
- **UnknownServiceException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()  
*Copy Constructor.*
- **UnknownServiceException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownServiceException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnknownServiceException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnknownServiceException** \* clone () const  
*Clones this exception.*
- virtual ~**UnknownServiceException** () throw ()

### 6.847.1 Constructor & Destructor Documentation

#### 6.847.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

#### 6.847.1.2 decaf::net::UnknownServiceException::UnknownServiceException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

### 6.847.1.3 `decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & ex) throw () [inline]`

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

### 6.847.1.4 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.847.1.5 `decaf::net::UnknownServiceException::UnknownServiceException (const std::exception * cause) throw () [inline]`

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.847.1.6 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message



**6.847.1.7**    **virtual**  
**decaf::net::UnknownServiceException::~UnknownServiceException ()**  
**throw ()**    [inline, virtual]

## 6.847.2 Member Function Documentation

**6.847.2.1**    **virtual UnknownServiceException\* de-**  
**caf::net::UnknownServiceException::clone () const**  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownServiceException.h**

## 6.848 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

#include <src/main/decaf/io/UnsupportedEncodingException.h> Inheritance diagram for decaf::io::UnsupportedEncodingException:

### Public Member Functions

- **UnsupportedEncodingException** () throw ()  
*Default Constructor.*
- **UnsupportedEncodingException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **UnsupportedEncodingException** (const **UnsupportedEncodingException** &ex) throw ()  
*Copy Constructor.*
- **UnsupportedEncodingException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnsupportedEncodingException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnsupportedEncodingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **UnsupportedEncodingException** \* clone () const  
*Clones this exception.*
- virtual ~**UnsupportedEncodingException** () throw ()

### 6.848.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since:

1.0

### 6.848.2 Constructor & Destructor Documentation

#### 6.848.2.1 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException () throw () [inline]

Default Constructor.

**6.848.2.2 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const lang::Exception & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* the exception to copy

**6.848.2.3 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & *ex*) throw () [inline]**

Copy Constructor.

**Parameters:**

*ex* the exception to copy, which is an instance of this type

**6.848.2.4 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ...) throw () [inline]**

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.848.2.5 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const std::exception \* *cause*) throw () [inline]**

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.848.2.6 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char \* *file*, const int *lineNumber*, const char \* *msg*, ...) throw () [inline]**

Constructor.

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

#### 6.848.2.7 virtual

```
decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException  
( ) throw ( ) [inline, virtual]
```

### 6.848.3 Member Function Documentation

#### 6.848.3.1 virtual UnsupportedEncodingException\* decaf::io::UnsupportedEncodingException::clone ( ) const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns:

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

## 6.849 decaf::lang::exceptions::UnsupportedOperationException Class Reference

#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h> Inheritance diagram for decaf::lang::exceptions::UnsupportedOperationException:

### Public Member Functions

- **UnsupportedOperationException** () throw ()  
*Default Constructor.*
- **UnsupportedOperationException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1831).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()  
*Copy Constructor.*
- **UnsupportedOperationException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnsupportedOperationException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnsupportedOperationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnsupportedOperationException** \* clone () const  
*Clones this exception.*
- virtual ~**UnsupportedOperationException** () throw ()

### 6.849.1 Constructor & Destructor Documentation

#### 6.849.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException () throw () [inline]

Default Constructor.

#### 6.849.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1831).

#### Parameters:

*ex* An exception that should become this type of **Exception** (p. 1831)

**6.849.1.3** `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException(const UnsupportedOperationException & ex) throw ()` [inline]

Copy Constructor.

**Parameters:**

*ex* An exception that should become this type of **Exception** (p. 1831)

**6.849.1.4** `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.849.1.5** `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException(const std::exception * cause) throw ()` [inline]

Constructor.

**Parameters:**

*cause* **Pointer** (p. 2946) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.849.1.6** `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException(const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.849.1.7** virtual  
decaf::lang::exceptions::UnsupportedOperationException::~~UnsupportedOperationException()  
() throw () [inline, virtual]

## 6.849.2 Member Function Documentation

**6.849.2.1** virtual UnsupportedOperationException\* decaf::lang::exceptions::UnsupportedOperationException::clone () const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

an new **Exception** (p. 1831) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1834).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 3171).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**UnsupportedOperationException.h**

## 6.850 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

#include <src/main/cms/UnsupportedOperationException.h> Inheritance diagram for cms::UnsupportedOperationException:

### Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

### 6.850.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since:

2.0

### 6.850.2 Constructor & Destructor Documentation

- 6.850.2.1 **cms::UnsupportedOperationException::UnsupportedOperationException** () throw ()
- 6.850.2.2 **cms::UnsupportedOperationException::UnsupportedOperationException** (const **UnsupportedOperationException** & *ex*) throw ()
- 6.850.2.3 **cms::UnsupportedOperationException::UnsupportedOperationException** (const std::string & *message*, const std::exception \* *cause*) throw ()
- 6.850.2.4 **cms::UnsupportedOperationException::UnsupportedOperationException** (const std::string & *message*, const std::exception \* *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.850.2.5 virtual **cms::UnsupportedOperationException::~~UnsupportedOperationException** () throw () [virtual]

The documentation for this class was generated from the following file:



- `src/main/cms/UnsupportedOperationException.h`

## 6.851 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 3921) as defined by RFC 2396.

#include <src/main/decaf/net/URI.h> Inheritance diagram for decaf::net::URI:

### Public Member Functions

- **URI** ()  
*Default Constructor, same as calling a Constructor with all fields empty.*
- **URI** (const **URI** &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) as a copy of another **URI** (p. 3921).*
- **URI** (const std::string &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3921) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **URI** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const

- `std::string getPath () const`
- `int getPort () const`
- `std::string getQuery () const`
- `std::string getScheme () const`
- `std::string getUserInfo () const`
- `std::string getRawAuthority () const`  
*Returns the raw authority component of this **URI** (p. 3921).*
- `std::string getRawFragment () const`  
*Returns the raw fragment component of this **URI** (p. 3921).*
- `std::string getRawPath () const`  
*Returns the raw path component of this **URI** (p. 3921).*
- `std::string getRawQuery () const`  
*Returns the raw query component of this **URI** (p. 3921).*
- `std::string getRawSchemeSpecificPart () const`  
*Returns the raw scheme-specific part of this **URI** (p. 3921).*
- `std::string getSchemeSpecificPart () const`  
*Returns the decoded scheme-specific part of this **URI** (p. 3921).*
- `std::string getRawUserInfo () const`  
*Returns the raw user-information component of this **URI** (p. 3921).*
- `bool isAbsolute () const`  
*Tells whether or not this **URI** (p. 3921) is absolute.*
- `bool isOpaque () const`  
*Tells whether or not this **URI** (p. 3921) is opaque.*
- `URI normalize () const`  
*Normalizes this **URI**'s path.*
- `URI parseServerAuthority () const throw ( URISyntaxException )`  
*Attempts to parse this **URI**'s authority component, if defined, into user-information, host, and port components.*
- `URI relativize (const URI &uri) const`  
*Relativizes the given **URI** (p. 3921) against this **URI** (p. 3921).*
- `URI resolve (const std::string &str) const throw ( lang::exceptions::IllegalArgumentException )`  
*Constructs a new **URI** (p. 3921) by parsing the given string and then resolving it against this **URI** (p. 3921).*
- `URI resolve (const URI &uri) const`  
*Resolves the given **URI** (p. 3921) against this **URI** (p. 3921).*

- `std::string toString () const`  
*Returns the content of this **URI** (p. 3921) as a string.*
- `URL toURL () const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )`  
*Constructs a **URL** (p. 3960) from this **URI** (p. 3921).*

## Static Public Member Functions

- `static URI create (const std::string uri) throw ( lang::exceptions::IllegalArgumentException )`  
*Creates a **URI** (p. 3921) by parsing the given string.*

### 6.851.1 Detailed Description

This class represents an instance of a **URI** (p. 3921) as defined by RFC 2396.

### 6.851.2 Constructor & Destructor Documentation

#### 6.851.2.1 `decaf::net::URI::URI ()`

Default Constructor, same as calling a Constructor with all fields empty.

#### 6.851.2.2 `decaf::net::URI::URI (const URI & uri) throw ( URISyntaxException )`

Constructs a **URI** (p. 3921) as a copy of another **URI** (p. 3921).

#### Parameters:

*uri* - uri to copy

#### 6.851.2.3 `decaf::net::URI::URI (const std::string & uri) throw ( URISyntaxException )`

Constructs a **URI** (p. 3921) from the given string.

#### Parameters:

*uri* - string uri to parse.

#### 6.851.2.4 `decaf::net::URI::URI (const std::string & scheme, const std::string & ssp, const std::string & fragment) throw ( URISyntaxException )`

Constructs a **URI** (p. 3921) from the given components.

#### Parameters:

*scheme* - the uri scheme

*ssp* - Scheme specific part

*fragment* - Fragment

**6.851.2.5** decaf::net::URI::URI (const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw ( URISyntaxException )

Constructs a **URI** (p. 3921) from the given components.

**Parameters:**

*scheme* - Scheme name

*userInfo* - User name and authorization information

*host* - Host name

*port* - Port number

*path* - Path

*query* - Query

*fragment* - Fragment

**6.851.2.6** decaf::net::URI::URI (const std::string & *scheme*, const std::string & *host*, const std::string & *path*, const std::string & *fragment*) throw ( URISyntaxException )

Constructs a **URI** (p. 3921) from the given components.

**Parameters:**

*scheme* - Scheme name

*host* - Host name

*path* - Path

*fragment* - Fragment

**6.851.2.7** decaf::net::URI::URI (const std::string & *scheme*, const std::string & *authority*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw ( URISyntaxException )

Constructs a **URI** (p. 3921) from the given components.

**Parameters:**

*scheme* - Scheme name

*authority* - Authority

*path* - Path

*query* - Query

*fragment* - Fragment

**6.851.2.8**    `virtual decaf::net::URI::~~URI ()` [inline, virtual]

### 6.851.3 Member Function Documentation

**6.851.3.1**    `virtual int decaf::net::URI::compareTo (const URI & value) const`  
[virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

**Parameters:**

*value* - the value to compare to this one.

**Returns:**

zero if equal minus one if less than and one if greater than.

**6.851.3.2**    `static URI decaf::net::URI::create (const std::string uri) throw (`  
`lang::exceptions::IllegalArgumentException )` [static]

Creates a **URI** (p. 3921) by parsing the given string. This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 3948) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

**Parameters:**

*uri* - **URI** (p. 3921) string to parse

**Exceptions:**

*IllegalArgumentException*

**6.851.3.3**    `virtual bool decaf::net::URI::equals (const URI & value) const` [virtual]

**Returns:**

true if this value is considered equal to the passed value.

**6.851.3.4**    `std::string decaf::net::URI::getAuthority () const`

**Returns:**

the decoded authority component of this **URI** (p. 3921).

**6.851.3.5**    `std::string decaf::net::URI::getFragment () const`

**Returns:**

the decoded fragment component of this **URI** (p. 3921).

**6.851.3.6    std::string decaf::net::URI::getHost () const****Returns:**

the host component of this **URI** (p. 3921).

**6.851.3.7    std::string decaf::net::URI::getPath () const****Returns:**

the path component of this **URI** (p. 3921).

**6.851.3.8    int decaf::net::URI::getPort () const****Returns:**

the port component of this **URI** (p. 3921).

**6.851.3.9    std::string decaf::net::URI::getQuery () const****Returns:**

the query component of this **URI** (p. 3921).

**6.851.3.10   std::string decaf::net::URI::getRawAuthority () const**

Returns the raw authority component of this **URI** (p. 3921). The authority component of a **URI** (p. 3921), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

**Returns:**

the raw authority component of the **URI** (p. 3921)

**6.851.3.11   std::string decaf::net::URI::getRawFragment () const**

Returns the raw fragment component of this **URI** (p. 3921). The fragment component of a **URI** (p. 3921), if defined, only contains legal **URI** (p. 3921) characters.

**Returns:**

the raw fragment component of this **URI** (p. 3921)

**6.851.3.12   std::string decaf::net::URI::getRawPath () const**

Returns the raw path component of this **URI** (p. 3921). The path component of a **URI** (p. 3921), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

**Returns:**

the raw path component of this **URI** (p. 3921)

**6.851.3.13** `std::string decaf::net::URI::getRawQuery () const`

Returns the raw query component of this **URI** (p. 3921). The query component of a **URI** (p. 3921), if defined, only contains legal **URI** (p. 3921) characters.

**Returns:**

the raw query component of the **URI** (p. 3921).

**6.851.3.14** `std::string decaf::net::URI::getRawSchemeSpecificPart () const`

Returns the raw scheme-specific part of this **URI** (p. 3921). The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 3921) only contains legal **URI** (p. 3921) characters.

**Returns:**

the raw scheme special part of the uri

**6.851.3.15** `std::string decaf::net::URI::getRawUserInfo () const`

Returns the raw user-information component of this **URI** (p. 3921). The user-information component of a **URI** (p. 3921), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

**Returns:**

the raw user-information component of the **URI** (p. 3921)

**6.851.3.16** `std::string decaf::net::URI::getScheme () const`**Returns:**

the scheme component of this **URI** (p. 3921)

**6.851.3.17** `std::string decaf::net::URI::getSchemeSpecificPart () const`

Returns the decoded scheme-specific part of this **URI** (p. 3921). The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

**Returns:**

the raw scheme specific part of the uri.



**6.851.3.18** `std::string decaf::net::URI::getUserInfo () const`**Returns:**

the user info component of this **URI** (p. 3921)

**6.851.3.19** `bool decaf::net::URI::isAbsolute () const`

Tells whether or not this **URI** (p. 3921) is absolute. A **URI** (p. 3921) is absolute if, and only if, it has a scheme component.

**Returns:**

true if, and only if, this **URI** (p. 3921) is absolute

**6.851.3.20** `bool decaf::net::URI::isOpaque () const`

Tells whether or not this **URI** (p. 3921) is opaque. A **URI** (p. 3921) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 3921) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

**Returns:**

true if, and only if, this **URI** (p. 3921) is opaque

**6.851.3.21** `URI decaf::net::URI::normalize () const`

Normalizes this **URI**'s path. If this **URI** (p. 3921) is opaque, or if its path is already in normal form, then this **URI** (p. 3921) is returned. Otherwise a new **URI** (p. 3921) is constructed that is identical to this **URI** (p. 3921) except that its path is computed by normalizing this **URI**'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed.
2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable.
3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 3921) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 3921) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

**Returns:**

A **URI** (p. 3921) equivalent to this **URI** (p. 3921), but whose path is in normal form

**6.851.3.22** `virtual bool decaf::net::URI::operator< (const URI & value) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.851.3.23** `virtual bool decaf::net::URI::operator==(const URI & value) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.851.3.24** `URI decaf::net::URI::parseServerAuthority () const throw (`  
`URISyntaxException )`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components. If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 3921) has no authority component, this method simply returns this **URI** (p. 3921).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

**Returns:**

A **URI** (p. 3921) whose authority field has been parsed as a server-based authority

**Exceptions:**

*URISyntaxException* (p. 3948) - If the authority component of this **URI** (p. 3921) is defined but cannot be parsed as a server-based authority.

**6.851.3.25** `URI decaf::net::URI::relativize (const URI & uri) const`

Relativizes the given **URI** (p. 3921) against this **URI** (p. 3921). The relativization of the given **URI** (p. 3921) against this **URI** (p. 3921) is computed as follows:

1. If either this **URI** (p. 3921) or the given **URI** (p. 3921) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 3921) is not a prefix of the path of the given **URI** (p. 3921), then the given **URI** (p. 3921) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 3921) is constructed with query and fragment components taken from the given **URI** (p. 3921) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

**Parameters:**

*uri* - The **URI** (p. 3921) to be relativized against this **URI** (p. 3921)

**Returns:**

The resulting **URI** (p. 3921)

**6.851.3.26 URI decaf::net::URI::resolve (const URI & uri) const**

Resolves the given **URI** (p. 3921) against this **URI** (p. 3921). If the given **URI** (p. 3921) is already absolute, or if this **URI** (p. 3921) is opaque, then a copy of the given **URI** (p. 3921) is returned.

If the given **URI**'s fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 3921) with the given fragment but with all other components equal to those of this **URI** (p. 3921) is returned. This allows a **URI** (p. 3921) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 3921).

Otherwise this method constructs a new hierarchical **URI** (p. 3921) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 3921) is constructed with this **URI**'s scheme and the given **URI**'s query and fragment components.
2. If the given **URI** (p. 3921) has an authority component then the new **URI**'s authority and path are taken from the given **URI** (p. 3921).
3. Otherwise the new **URI**'s authority component is copied from this **URI** (p. 3921), and its path is computed as follows:

1. If the given **URI**'s path is absolute then the new **URI**'s path is taken from the given **URI** (p. 3921).
2. Otherwise the given **URI**'s path is relative, and so the new **URI**'s path is computed by resolving the path of the given **URI** (p. 3921) against the path of this **URI** (p. 3921). This is done by concatenating all but the last segment of this **URI**'s path, if any, with the given **URI**'s path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 3921) is absolute or the given **URI** (p. 3921) is absolute.

**Parameters:**

*uri* - The **URI** (p. 3921) to be resolved against this **URI** (p. 3921)

**Returns:**

The resulting **URI** (p. 3921)

**6.851.3.27 URI decaf::net::URI::resolve (const std::string & str) const throw ( lang::exceptions::IllegalArgumentException )**

Constructs a new **URI** (p. 3921) by parsing the given string and then resolving it against this **URI** (p. 3921). This convenience method works as if invoking it were equivalent to evaluating the expression `resolve( URI::create( str ) )`.

**Parameters:**

*str* - The string to be parsed into a **URI** (p. 3921)

**Returns:**

The resulting **URI** (p. 3921)

**Exceptions:**

***IllegalArgumentException*** - If the given string violates RFC 2396

**6.851.3.28** `std::string decaf::net::URI::toString () const`

Returns the content of this **URI** (p. 3921) as a string. If this **URI** (p. 3921) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 3921) was created by normalization, resolution, or relativization, and so a string is constructed from this **URI**'s components according to the rules specified in RFC 2396, section 5.2, step 7.

**Returns:**

the string form of this **URI** (p. 3921)

**6.851.3.29** `URL decaf::net::URI::toURL () const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )`

Constructs a **URL** (p. 3960) from this **URI** (p. 3921). This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 3960)(this.toString())` after first checking that this **URI** (p. 3921) is absolute.

**Returns:**

A **URL** (p. 3960) constructed from this **URI** (p. 3921)

**Exceptions:**

***IllegalArgumentException*** - If this **URL** (p. 3960) is not absolute

***MalformedURLException*** (p. 2456) - If a protocol handler for the **URL** (p. 3960) could not be found, or if some other error occurred while constructing the **URL** (p. 3960)

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URI.h`

## 6.852 decaf::internal::net::URIEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URIEncoderDecoder.h>
```

### Public Member Functions

- **URIEncoderDecoder** ()
- virtual **~URIEncoderDecoder** ()

### Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:.*
- static void **validateSimple** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:.*
- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)  
*All characters except letters ('a').*
- static std::string **encodeOthers** (const std::string &s)  
*Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.*
- static std::string **decode** (const std::string &s)  
*Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.*

### 6.852.1 Constructor & Destructor Documentation

**6.852.1.1** decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ()

**6.852.1.2** virtual decaf::internal::net::URIEncoderDecoder::~~URIEncoderDecoder () [inline, virtual]

### 6.852.2 Member Function Documentation

**6.852.2.1** static std::string decaf::internal::net::URIEncoderDecoder::decode (const std::string & s) [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme. " and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

**Parameters:**

*s* - The encoded string.

**Returns:**

The decoded version.

**6.852.2.2** `static std::string decaf::internal::net::URIEncoderDecoder::encodeOthers  
(const std::string & s) [static]`

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved. They are converted into their hexadecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

**Parameters:**

*s* - the string to be converted

**Returns:**

the converted string

**6.852.2.3** `static std::string decaf::internal::net::URIEncoderDecoder::quoteIllegal  
(const std::string & s, const std::string & legal) [static]`

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ”.

For example: '#' -> 23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

**Parameters:**

*s* - the string to be converted

*legal* - the characters allowed to be preserved in the string *s*

**Returns:**

converted string

**6.852.2.4** `static void decaf::internal::net::URIEncoderDecoder::validate  
(const std::string & s, const std::string & legal) throw (  
decaf::net::URISyntaxException ) [static]`

Validate a string by checking if it contains any characters other than: 1. letters ('a'..'z', 'A'..'Z') 2. numbers ('0'..'9') 3. characters in the legalset parameter 4. characters that are not ISO Control or are not ISO Space characters)

**Parameters:**

*s* - the string to be validated

*legal* - the characters allowed in the string *s*

**6.852.2.5** static void decaf::internal::net::URIEncoderDecoder::validateSimple  
(const std::string & *s*, const std::string & *legal*) throw (  
decaf::net::URISyntaxException ) [static]

Validate a string by checking if it contains any characters other than: 1. letters ('a'..'z', 'A'..'Z')  
2. numbers ('0'..'9') 3. characters in the legalset parameter

**Parameters:**

- s* - the string to be validated
- legal* - the characters allowed in the string *s*

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIEncoderDecoder.h**

## 6.853 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

### Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)

*Setup the **URIHelper** (p. 3935) with values assigned to the various fields that are used in the validation process.*

- **URIHelper** ()

*Sets up the filter strings with sane defaults.*

- virtual ~**URIHelper** ()

- **URIType parseURI** (const std::string &uri, bool forceServer) throw ( decaf::net::URISyntaxException )

*Parse the passed in URI.*

- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw ( decaf::net::URISyntaxException )

*Validate the schema portin of the URI.*

- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size\_t index) throw ( decaf::net::URISyntaxException )

*Validate that the URI Ssp Segment contains no invalid encodings.*

- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size\_t index) throw ( decaf::net::URISyntaxException )

*Validate that the URI Authority Segment contains no invalid encodings.*

- void **validatePath** (const std::string &uri, const std::string &path, std::size\_t index) throw ( decaf::net::URISyntaxException )

*Validate that the URI Path Segment contains no invalid encodings.*

- void **validateQuery** (const std::string &uri, const std::string &query, std::size\_t index) throw ( decaf::net::URISyntaxException )

*Validate that the URI Query Segment contains no invalid encodings.*

- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size\_t index) throw ( decaf::net::URISyntaxException )

*Validate that the URI fragment contains no invalid encodings.*

- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw ( decaf::net::URISyntaxException )

*determine the host, port and user-info if the authority parses successfully to a server based authority*



- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Check the supplied user info for validity.*
- bool **isValidHost** (bool forceServer, const std::string &host) throw ( decaf::net::URISyntaxException )  
*distinguish between IPv4, IPv6, domain name and validate it based on its type*
- bool **isValidDomainName** (const std::string &host)  
*Validates the string past to determine if it is a well formed domain name.*
- bool **isValidIPv4Address** (const std::string &host)  
*Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.*
- bool **isValidIPv6Address** (const std::string &ipAddress)  
*Determines if the given address is valid according to the IPv6 spec.*
- bool **isValidIPv4Word** (const std::string &word)  
*Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.*
- bool **isValidHexChar** (char c)  
*Determines if the given char is a valid Hex char.*

### 6.853.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

### 6.853.2 Constructor & Destructor Documentation

#### 6.853.2.1 decaf::internal::net::URIHelper::URIHelper (const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal*)

Setup the **URIHelper** (p. 3935) with values assigned to the various fields that are used in the validation process. The defaults are overridden by these values.

##### Parameters:

- unreserved* - characters not reserved for use.
- punct* - allowable punctuation symbols.
- reserved* - characters not allowed for general use in the URI.
- someLegal* - characters that are legal in certain cases.
- allLegal* - characters that are always legal.

#### 6.853.2.2 decaf::internal::net::URIHelper::URIHelper ()

Sets up the filter strings with sane defaults.

**6.853.2.3** `virtual decaf::internal::net::URIHelper::~~URIHelper () [inline, virtual]`

### 6.853.3 Member Function Documentation

**6.853.3.1** `bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & host)`

Validates the string past to determine if it is a well formed domain name.

**Parameters:**

*host* - domain name to validate.

**Returns:**

true if host is well formed.

**6.853.3.2** `bool decaf::internal::net::URIHelper::isValidHexChar (char c)`

Determines if the given char is a valid Hex char. Valid chars are A-F (upper or lower case) and 0-9.

**Parameters:**

*c* - char to inspect

**Returns:**

true if *c* is a valid hex char.

**6.853.3.3** `bool decaf::internal::net::URIHelper::isValidHost (bool forceServer, const std::string & host) throw ( decaf::net::URISyntaxException )`

distinguish between IPv4, IPv6, domain name and validate it based on its type

**Parameters:**

*forceServer* - true if the forceServer mode should be active.

*host* - Host string to validate.

**Returns:**

true if the host value if a valid domain name.

**Exceptions:**

*URISyntaxException* if the host is invalid and forceServer is true.

**6.853.3.4** `bool decaf::internal::net::URIHelper::isValidIP4Word (const std::string & word)`

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

**Parameters:**

*word* - string value to check.

**Returns:**

true if the word is a valid IPv4 word.

**6.853.3.5    bool decaf::internal::net::URIHelper::isValidIPv6Address (const std::string & *ipAddress*)**

Determines if the given address is valid according to the IPv6 spec.

**Parameters:**

*ipAddress* - string ip address value to validate.

**Returns:**

true if the address string is valid.

**6.853.3.6    bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & *host*)**

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9. and XXX is not greater than 255.

**Parameters:**

*host* - IPv4 address string to parse.

**Returns:**

true if host is a well formed IPv4 address.

**6.853.3.7    URIType decaf::internal::net::URIHelper::parseAuthority (bool *forceServer*, const std::string & *authority*) throw ( decaf::net::URISyntaxException )**

determine the host, port and user-info if the authority parses successfully to a server based authority behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

**Parameters:**

*forceServer*  
*authority*

**Returns:**

a URIType (p. 3952) instance containing the parsed data.

**Exceptions:**

*URISyntaxException*

**6.853.3.8**    **URIType** decaf::internal::net::URIHelper::parseURI (const std::string & *uri*, bool *forceServer*) throw ( decaf::net::URISyntaxException )

Parse the passed in URI.

**Parameters:**

*uri* - the URI to Parse

*forceServer* - if true invalid URI data throws an Exception

**Returns:**

a **URIType** (p. 3952) instance containing the parsed data.

**Exceptions:**

*URISyntaxException* if forceServer is true and the URI is invalid.

**6.853.3.9**    **void** decaf::internal::net::URIHelper::validateAuthority (const std::string & *uri*, const std::string & *authority*, std::size\_t *index*) throw ( decaf::net::URISyntaxException )

Validate that the URI Authority Segment contains no invalid encodings.

**Parameters:**

*uri* - the full uri.

*authority* - the Authority to check.

*index* - position in the uri where Authority starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.10**    **void** decaf::internal::net::URIHelper::validateFragment (const std::string & *uri*, const std::string & *fragment*, std::size\_t *index*) throw ( decaf::net::URISyntaxException )

Validate that the URI fragment contains no invalid encodings.

**Parameters:**

*uri* - the full uri.

*fragment* - the fragment to check.

*index* - position in the uri where fragment starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.11** void decaf::internal::net::URIHelper::validatePath (const std::string & *uri*, const std::string & *path*, std::size\_t *index*) throw ( decaf::net::URISyntaxException )

Validate that the URI Path Segment contains no invalid encodings.

**Parameters:**

*uri* - the full uri.

*path* - the path to check.

*index* - position in the uri where path starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.12** void decaf::internal::net::URIHelper::validateQuery (const std::string & *uri*, const std::string & *query*, std::size\_t *index*) throw ( decaf::net::URISyntaxException )

Validate that the URI Query Segment contains no invalid encodings.

**Parameters:**

*uri* - the full uri.

*query* - the query to check.

*index* - position in the uri where fragment starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.13** void decaf::internal::net::URIHelper::validateScheme (const std::string & *uri*, const std::string & *scheme*, int *index*) throw ( decaf::net::URISyntaxException )

Validate the schema portin of the URI.

**Parameters:**

*uri* - the URI to check.

*scheme* - the schema section of the URI.

*index* - index in uri where schema starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.14**   `void decaf::internal::net::URIHelper::validateSsp (const std::string & uri, const std::string & ssp, std::size_t index) throw ( decaf::net::URISyntaxException )`

Validate that the URI Ssp Segment contains no invalid encodings.

**Parameters:**

*uri* - the full uri.

*ssp* - the SSP to check.

*index* - position in the uri where Ssp starts.

**Exceptions:**

*URISyntaxException* if the fragment has errors.

**6.853.3.15**   `void decaf::internal::net::URIHelper::validateUserinfo (const std::string & uri, const std::string & userinfo, std::size_t index) throw ( decaf::net::URISyntaxException )`

Check the supplied user info for validity.

**Parameters:**

*uri* - the uri to parse.

*userinfo* - supplied user info

*index* - index into the URI string where the data is located.

**Returns:**

true if valid

**Exceptions:**

*URISyntaxException* if an error occurs

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIHelper.h`

## 6.854 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

### Public Member Functions

- **URIPool** ()  
*Create an Empty URI Pool.*
- **URIPool** (const **decaf::util::List**< **URI** > &uris)  
*Creates a new URI Pool using the given list as the initial Free List.*
- **~URIPool** ()
- **URI** **getURI** () throw ( **decaf::lang::exceptions::NoSuchElementException** )  
*Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`.*
- void **addURI** (const **URI** &uri)  
*Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.*
- void **addURIs** (const **StlList**< **URI** > &uris)  
*Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.*
- void **removeURI** (const **URI** &uri)  
*Remove a given URI from the Free List.*
- bool **isRandomize** () const  
*Is the URI that is given randomly picked from the pool or is each one taken in sequence.*
- void **setRandomize** (bool value)  
*Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.*

### 6.854.1 Constructor & Destructor Documentation

#### 6.854.1.1 activemq::transport::failover::URIPool::URIPool ()

Create an Empty URI Pool.

#### 6.854.1.2 activemq::transport::failover::URIPool::URIPool (const **decaf::util::List**< **URI** > & *uris*)

Creates a new URI Pool using the given list as the initial Free List.

#### Parameters:

*uris* - List of URI to place in the Pool.

**6.854.1.3    `activemq::transport::failover::URIPool::~~URIPool ()`****6.854.2    Member Function Documentation****6.854.2.1    `void activemq::transport::failover::URIPool::addURI (const URI & uri)`**

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

**Parameters:**

*uri* - a URI previously taken from the pool.

**6.854.2.2    `void activemq::transport::failover::URIPool::addURIs (const StlList< URI > & uris)`**

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

**Parameters:**

*uris* - List of URIs to add into the Pool.

**6.854.2.3    `URI activemq::transport::failover::URIPool::getURI () throw ( decaf::lang::exceptions::NoSuchElementException )`**

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`. Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

**Returns:**

the next free URI in the Pool.

**Exceptions:**

*NoSuchElementException* if there are none free currently.

**6.854.2.4    `bool activemq::transport::failover::URIPool::isRandomize () const [inline]`**

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

**Returns:**

true if URI gets are random.

**6.854.2.5    `void activemq::transport::failover::URIPool::removeURI (const URI & uri)`**

Remove a given URI from the Free List.



**Parameters:**

*uri* - the URI to find and remove from the free list

**6.854.2.6 void activemq::transport::failover::URIPool::setRandomize (bool *value*) [inline]**

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

**Parameters:**

*value* - true indicates URI gets are random.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**URIPool.h**

## 6.855 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

### Static Public Member Functions

- static void **parseURL** (const std::string &URI, **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::IllegalArgumentException** )  
*Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.*
- static **CompositeData** **parseComposite** (const **URI** &uri) throw ( **decaf::net::URISyntaxException** )  
*Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.*
- static **decaf::util::Properties** **parseQuery** (std::string query) throw ( **decaf::lang::exceptions::IllegalArgumentException** )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*
- static void **parseQuery** (std::string query, **decaf::util::Properties** \*properties) throw ( **decaf::lang::exceptions::IllegalArgumentException** )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*
- static std::string **createQueryString** (const **Properties** &options) throw ( **decaf::net::URISyntaxException** )  
*Given a properties object create a string that can be appended to a URI as a valid Query string.*

### 6.855.1 Member Function Documentation

**6.855.1.1** static std::string **activemq::util::URISupport::createQueryString** (const **Properties** & *options*) throw ( **decaf::net::URISyntaxException** )  
 [static]

Given a properties object create a string that can be appended to a URI as a valid Query string.

#### Parameters:

*options* Properties object containing key / value query values.

#### Returns:

a valid URI query string.

#### Exceptions:

**URISyntaxException** if the string in the Properties object can't be encoded into a valid URI Query string.

### 6.855.1.2 static CompositeData activemq::util::URISupport::parseComposite (const URI & *uri*) throw ( decaf::net::URISyntaxException ) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

#### Parameters:

*uri* - The Composite URI to parse.

#### Returns:

a new **CompositeData** (p. 1219) object with the parsed data

#### Exceptions:

*URISyntaxException* if the URI is not well formed.

### 6.855.1.3 static void activemq::util::URISupport::parseQuery (std::string *query*, decaf::util::Properties \* *properties*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

#### Parameters:

*query* - the query string to parse.

*properties* - object pointer to get the parsed output.

#### Exceptions:

*IllegalArgumentException* if the Query string is not well formed.

### 6.855.1.4 static decaf::util::Properties activemq::util::URISupport::parseQuery (std::string *query*) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

#### Parameters:

*query* The query string to parse and extract the encoded properties.

#### Returns:

Properties object with the parsed output.

#### Exceptions:

*IllegalArgumentException* if the Query string is not well formed.

**6.855.1.5** `static void activemq::util::URISupport::parseURL (const  
std::string & URI, decaf::util::Properties & properties) throw (  
decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

**Parameters:**

*URI* a Broker URI to parse

*properties* a Properties object to set the parsed values in

**Exceptions:**

*IllegalArgumentException* if the passed URI is invalid

The documentation for this class was generated from the following file:

- `src/main/activemq/util/URISupport.h`

## 6.856 decaf::net::URISyntaxException Class Reference

#include <src/main/decaf/net/URISyntaxException.h> Inheritance diagram for decaf::net::URISyntaxException:

### Public Member Functions

- **URISyntaxException** () throw ()  
*Default Constructor.*
- **URISyntaxException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()  
*Copy Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const std::exception \*cause) throw ()  
*Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const char \*msg DECAF\_ - UNUSED) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason, int index) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **URISyntaxException** \* **clone** () const  
*Clones this exception.*
- virtual ~**URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- int **getIndex** () const

### 6.856.1 Constructor & Destructor Documentation

#### 6.856.1.1 decaf::net::URISyntaxException::URISyntaxException () throw () [inline]

Default Constructor.

### 6.856.1.2 `decaf::net::URISyntaxException::URISyntaxException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.856.1.3 `decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & ex) throw () [inline]`

Copy Constructor.

#### Parameters:

*ex* An exception that should become this type of Exception

### 6.856.1.4 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters:

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

### 6.856.1.5 `decaf::net::URISyntaxException::URISyntaxException (const std::exception * cause) throw () [inline]`

Constructor.

#### Parameters:

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

### 6.856.1.6 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const char *msg DECAF_UNUSED) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs  
*lineNumber* The line number where the exception occurred.  
*msg* The message to report  
... list of primitives that are formatted into the message

**6.856.1.7** `decaf::net::URISyntaxException::URISyntaxException (const char * file,  
const int lineNumber, const std::string & input, const std::string & reason) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

**Parameters:**

*file* The file name where exception occurs.  
*lineNumber* The line number where the exception occurred.  
*input* The **URL** (p.3960) that caused the exception.  
*reason* The reason for the failure.

**6.856.1.8** `decaf::net::URISyntaxException::URISyntaxException (const char * file,  
const int lineNumber, const std::string & input, const std::string & reason, int index) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

**Parameters:**

*file* The file name where exception occurs  
*lineNumber* The line number where the exception occurred.  
*input* The input **URI** (p.3921) that caused the exception  
*reason* The reason for the failure.  
*index* The index in the **URI** (p.3921) string where the error occurred.

**6.856.1.9** `virtual decaf::net::URISyntaxException::~~URISyntaxException () throw  
()` [inline, virtual]

**6.856.2 Member Function Documentation**

**6.856.2.1** `virtual URISyntaxException* decaf::net::URISyntaxException::clone ()  
const` [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns:**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p.1834).

**6.856.2.2** `int decaf::net::URISyntaxException::getIndex () const [inline]`

**Returns:**

the index in the input string where the error occurred or -1

**6.856.2.3** `std::string decaf::net::URISyntaxException::getInput () const [inline]`

**Returns:**

the Input string that cause this exception or ""

**6.856.2.4** `std::string decaf::net::URISyntaxException::getReason () const [inline]`

**Returns:**

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`



## 6.857 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

### Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const  
*Gets the source URI string that was parsed to obtain this **URIType** (p. 3952) instance and the resulting data,.*
- void **setSource** (const std::string &source)  
*Sets the source URI string that was parsed to obtain this **URIType** (p. 3952) instance and the resulting data,.*
- std::string **getScheme** () const  
*Gets the Scheme of the URI, e.g.*
- void **setScheme** (const std::string &scheme)  
*Sets the Scheme of the URI, e.g.*
- std::string **getSchemeSpecificPart** () const  
*Gets the Scheme Specific Part of the URI.*
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)  
*Sets the Scheme Specific Part of the URI.*
- std::string **getAuthority** () const  
*Gets the Authority of the URI.*
- void **setAuthority** (const std::string &authority)  
*Sets the Authority of the URI.*
- std::string **getUserInfo** () const  
*Gets the user info part of the URI, e.g.*
- void **setUserInfo** (const std::string &userinfo)  
*Sets the user info part of the URI, e.g.*
- std::string **getHost** () const  
*Gets the Host name part of the URI.*
- void **setHost** (const std::string &host)  
*Sets the Host name part of the URI.*
- int **getPort** () const

*Gets the port part of the URI.*

- void **setPort** (int port)  
*Sets the port part of the URI.*
- std::string **getPath** () const  
*Gets the Path part of the URI.*
- void **setPath** (const std::string &path)  
*Sets the Path part of the URI.*
- std::string **getQuery** () const  
*Gets the Query part of the URI.*
- void **setQuery** (const std::string &query)  
*Sets the Query part of the URI.*
- std::string **getFragment** () const  
*Gets the Fragment part of the URI.*
- void **setFragment** (const std::string &fragment)  
*Sets the Fragment part of the URI.*
- bool **isOpaque** () const  
*Gets if the URI is Opaque.*
- void **setOpaque** (bool opaque)  
*Sets if the URI is Opaque.*
- bool **isAbsolute** () const  
*Gets if the URI is Absolute.*
- void **setAbsolute** (bool absolute)  
*Sets if the URI is Absolute.*
- bool **isServerAuthority** () const  
*Gets if the URI is a Server Authority.*
- void **setServerAuthority** (bool serverAuthority)  
*Sets if the URI is a Server Authority.*
- bool **isValid** () const  
*Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*
- void **setValid** (bool valid)  
*Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*

## 6.857.1 Detailed Description

Basic type object that holds data that composes a given URI.

## 6.857.2 Constructor & Destructor Documentation

**6.857.2.1** `decaf::internal::net::URIType::URIType (const std::string & source)` `[inline]`

**6.857.2.2** `decaf::internal::net::URIType::URIType ()` `[inline]`

**6.857.2.3** `virtual decaf::internal::net::URIType::~~URIType ()` `[inline, virtual]`

## 6.857.3 Member Function Documentation

**6.857.3.1** `std::string decaf::internal::net::URIType::getAuthority () const` `[inline]`

Gets the Authority of the URI.

### Returns:

Authority part string.

**6.857.3.2** `std::string decaf::internal::net::URIType::getFragment () const` `[inline]`

Gets the Fragment part of the URI.

### Returns:

Fragment part string.

**6.857.3.3** `std::string decaf::internal::net::URIType::getHost () const` `[inline]`

Gets the Host name part of the URI.

### Returns:

Host name part string.

**6.857.3.4** `std::string decaf::internal::net::URIType::getPath () const` `[inline]`

Gets the Path part of the URI.

### Returns:

Path part string.

**6.857.3.5** `int decaf::internal::net::URIType::getPort () const [inline]`

Gets the port part of the URL.

**Returns:**

port part string, -1 if not set.

**6.857.3.6** `std::string decaf::internal::net::URIType::getQuery () const [inline]`

Gets the Query part of the URL.

**Returns:**

Query part string.

**6.857.3.7** `std::string decaf::internal::net::URIType::getScheme () const [inline]`

Gets the Scheme of the URL, e.g. scheme ("http"/"ftp"/...).

**Returns:**

scheme part string.

**6.857.3.8** `std::string decaf::internal::net::URIType::getSchemeSpecificPart () const [inline]`

Gets the Scheme Specific Part of the URL.

**Returns:**

scheme specific part string.

**6.857.3.9** `std::string decaf::internal::net::URIType::getSource () const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p. 3952) instance and the resulting data,.

**Returns:**

the source URI string

**6.857.3.10** `std::string decaf::internal::net::URIType::getUserInfo () const [inline]`

Gets the user info part of the URL, e.g. user name, as in `http://user:passwd@host:port/`

**Returns:**

user info part string.

**6.857.3.11** `bool decaf::internal::net::URIType::isAbsolute () const [inline]`

Gets if the URI is Absolute.

**Returns:**

true if Absolute.

**6.857.3.12** `bool decaf::internal::net::URIType::isOpaque () const [inline]`

Gets if the URI is Opaque.

**Returns:**

true if opaque.

**6.857.3.13** `bool decaf::internal::net::URIType::isServerAuthority () const [inline]`

Gets if the URI is a Server Authority.

**Returns:**

true if Server Authority.

**6.857.3.14** `bool decaf::internal::net::URIType::isValid () const [inline]`

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

**Returns:**

true if the **URIType** (p. 3952) contains valid data.

**6.857.3.15** `void decaf::internal::net::URIType::setAbsolute (bool absolute) [inline]`

Sets if the URI is Absolute.

**Parameters:**

*absolute* - true if Absolute.

**6.857.3.16** `void decaf::internal::net::URIType::setAuthority (const std::string &authority) [inline]`

Sets the Authority of the URI.

**Parameters:**

*authority* Authority part string.

**6.857.3.17**   `void decaf::internal::net::URIType::setFragment (const std::string & fragment)` [inline]

Sets the Fragment part of the URI.

**Parameters:**

*fragment* - Fragment part string.

**6.857.3.18**   `void decaf::internal::net::URIType::setHost (const std::string & host)` [inline]

Sets the Host name part of the URI.

**Parameters:**

*host* - Host name part string.

**6.857.3.19**   `void decaf::internal::net::URIType::setOpaque (bool opaque)` [inline]

Sets if the URI is Opaque.

**Parameters:**

*opaque* true if opaque.

**6.857.3.20**   `void decaf::internal::net::URIType::setPath (const std::string & path)` [inline]

Sets the Path part of the URI.

**Parameters:**

*path* - Path part string.

**6.857.3.21**   `void decaf::internal::net::URIType::setPort (int port)` [inline]

Sets the port part of the URI.

**Parameters:**

*port* - port part string, -1 if not set.

**6.857.3.22**   `void decaf::internal::net::URIType::setQuery (const std::string & query)` [inline]

Sets the Query part of the URI.

**Parameters:**

*query* - Query part string.

**6.857.3.23** void decaf::internal::net::URIType::setScheme (const std::string & *scheme*) [inline]

Sets the Scheme of the URI, e.g. scheme ("http"/"ftp"/...).

**Parameters:**

*scheme* - scheme part string.

**6.857.3.24** void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & *schemeSpecificPart*) [inline]

Sets the Scheme Specific Part of the URI.

**Parameters:**

*schemeSpecificPart* - scheme specific part string.

**6.857.3.25** void decaf::internal::net::URIType::setServerAuthority (bool *serverAuthority*) [inline]

Sets if the URI is a Server Authority.

**Parameters:**

*serverAuthority* - true if Server Authority.

**6.857.3.26** void decaf::internal::net::URIType::setSource (const std::string & *source*) [inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 3952) instance and the resulting data,.

**Parameters:**

*source* - the source URI string

**6.857.3.27** void decaf::internal::net::URIType::setUserInfo (const std::string & *userinfo*) [inline]

Sets the user info part of the URI, e.g. user name, as in http://user:passwd@host:port/

**Parameters:**

*userinfo* - user info part string.

**6.857.3.28** void decaf::internal::net::URIType::setValid (bool *valid*) [inline]

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

**Parameters:**

*valid* - true if the **URIType** (p. 3952) contains valid data.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`



## 6.858 decaf::net::URL Class Reference

Class **URL** (p. 3960) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

### Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

### 6.858.1 Detailed Description

Class **URL** (p. 3960) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

```
http://www.ksc.nasa.gov/facts/internet/url-primer.html
```

In general, a **URL** (p. 3960) can be broken into several parts. The previous example of a **URL** (p. 3960) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named www.ksc.nasa.gov. The information on that host machine is named /facts/internet/url-primer.html. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 3960) is called the path component.

A **URL** (p. 3960) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 3960) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 3921)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports scope\_ids. The syntax and usage of scope\_ids is described here.

A **URL** (p. 3960) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 3960). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag chapter1 attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 3960). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 3960):

```
http://www.apache.org/cms/index.html
```

contained within it the relative **URL** (p. 3960):

FAQ.html

it would be a shorthand for:

`http://www.apache.org/cms/FAQ.html`

The relative **URL** (p. 3960) need not specify all the components of a **URL** (p. 3960). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 3960). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 3960) class does not itself encode or decode any **URL** (p. 3960) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 3960), and also to decode any escaped fields, that are returned from **URL** (p. 3960). Furthermore, because **URL** (p. 3960) has no knowledge of **URL** (p. 3960) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 3960). For example, the two URLs:

`http://foo.com/hello world/` and `http://foo.com/hello%20world`

would be considered not equal to each other.

Note, the **URI** (p. 3921) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 3921), and to convert between these two classes using `toURI()` and `URI.toURL()` (p. 3931).

The **URLEncoder** (p. 3963) and **URLDecoder** (p. 3962) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

## 6.858.2 Constructor & Destructor Documentation

### 6.858.2.1 `decaf::net::URL::URL ()`

### 6.858.2.2 `decaf::net::URL::URL (const std::string & url)`

### 6.858.2.3 `virtual decaf::net::URL::~~URL ()` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

## 6.859 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

### Public Member Functions

- virtual `~URLDecoder ()`

### Static Public Member Functions

- static `std::string decode (const std::string &value)`  
*Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type.*

### 6.859.1 Constructor & Destructor Documentation

**6.859.1.1** virtual `decaf::net::URLDecoder::~~URLDecoder ()` [inline, virtual]

### 6.859.2 Member Function Documentation

**6.859.2.1** static `std::string decaf::net::URLDecoder::decode (const std::string &value)` [static]

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type. '+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

#### Parameters:

*value* - string The encoded string.

#### Returns:

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

## 6.860 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

### Public Member Functions

- virtual `~URLEncoder()`

### Static Public Member Functions

- static `std::string encode(const std::string &value)`

*This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.*

### 6.860.1 Constructor & Destructor Documentation

**6.860.1.1** virtual `decaf::net::URLEncoder::~~URLEncoder()` [inline, virtual]

### 6.860.2 Member Function Documentation

**6.860.2.1** static `std::string decaf::net::URLEncoder::encode(const std::string &value)` [static]

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type. All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '\*', '\_' are converted into their hexadecimal value prepended by ".

For example: '#' -> 23

In addition, spaces are substituted by '+'

#### Parameters:

*value* - the string to be converted

#### Returns:

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

## 6.861 activemq::util::Usage Class Reference

#include <src/main/activemq/util/Usage.h> Inheritance diagram for activemq::util::Usage:

### Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`  
*Waits forever for more space to be returned to this **Usage** (p. 3964) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`  
*Waits for more space to be returned to this **Usage** (p. 3964) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`  
*Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void `increaseUsage (unsigned long long value)=0`  
*Increases the usage by the value amount.*
- virtual void `decreaseUsage (unsigned long long value)=0`  
*Decreases the usage by the value amount.*
- virtual bool `isFull () const =0`  
*Returns true if this **Usage** (p. 3964) instance is full, i.e.*

### 6.861.1 Constructor & Destructor Documentation

**6.861.1.1** virtual `activemq::util::Usage::~~Usage ()` [inline, virtual]

### 6.861.2 Member Function Documentation

**6.861.2.1** virtual void `activemq::util::Usage::decreaseUsage (unsigned long long value)` [pure virtual]

Decreases the usage by the value amount.

#### Parameters:

*value* Amount of space to return to the pool

Implemented in `activemq::util::MemoryUsage` (p. 2513).

**6.861.2.2** virtual void `activemq::util::Usage::enqueueUsage (unsigned long long value)` [pure virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

**Parameters:**

*value* Amount of usage in bytes to add.

Implemented in **activemq::util::MemoryUsage** (p. 2513).

**6.861.2.3 virtual void activemq::util::Usage::increaseUsage (unsigned long long *value*) [pure virtual]**

Increases the usage by the value amount.

**Parameters:**

*value* Amount of usage to add.

Implemented in **activemq::util::MemoryUsage** (p. 2514).

**6.861.2.4 virtual bool activemq::util::Usage::isFull () const [pure virtual]**

Returns true if this **Usage** (p. 3964) instance is full, i.e. **Usage** (p. 3964)  $\geq 100\%$

**Returns:**

true if **Usage** (p. 3964) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 2514).

**6.861.2.5 virtual void activemq::util::Usage::waitForSpace (unsigned int *timeout*) [pure virtual]**

Waits for more space to be returned to this **Usage** (p. 3964) Manager, times out when the given time span in milliseconds elapses.

**Parameters:**

*timeout* The time to wait for more space.

Implemented in **activemq::util::MemoryUsage** (p. 2514).

**6.861.2.6 virtual void activemq::util::Usage::waitForSpace () [pure virtual]**

Waits forever for more space to be returned to this **Usage** (p. 3964) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 2515).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/Usage.h`

## 6.862 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

#include <src/main/decaf/io/UTFDataFormatException.h>Inheritance diagram for decaf::io::UTFDataFormatException:

### Public Member Functions

- **UTFDataFormatException** () throw ()  
*Default Constructor.*
- **UTFDataFormatException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **UTFDataFormatException** (const UTFDataFormatException &ex) throw ()  
*Copy Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UTFDataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **UTFDataFormatException \* clone** () const  
*Clones this exception.*
- virtual ~**UTFDataFormatException** () throw ()

### 6.862.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since:

1.0

### 6.862.2 Constructor & Destructor Documentation

#### 6.862.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException () throw () [inline]

Default Constructor.

**6.862.2.2** `decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

**Parameters:**

*ex* the exception to copy

**6.862.2.3** `decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & ex) throw () [inline]`

Copy Constructor.

**Parameters:**

*ex* the exception to copy, which is an instance of this type

**6.862.2.4** `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.862.2.5** `decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * cause) throw () [inline]`

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.862.2.6** `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

**Parameters:**

*file* The file name where exception occurs



*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.862.2.7** virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException  
( ) throw ( ) [inline, virtual]

## 6.862.3 Member Function Documentation

**6.862.3.1** virtual UTFDataFormatException\* decaf::io::UTFDataFormatException::clone ( ) const  
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- src/main/decaf/io/UTFDataFormatException.h

## 6.863 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 3969)).

#include <src/main/decaf/util/UUID.h> Inheritance diagram for decaf::util::UUID:

### Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)  
*Constructs a new **UUID** (p. 3969) using the specified data.*
- virtual ~**UUID** ()
- virtual int **compareTo** (const **UUID** &value) const  
*Compare the given **UUID** (p. 3969) to this one.*
- virtual bool **equals** (const **UUID** &value) const  
*Compares this **UUID** (p. 3969) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **UUID** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual std::string **toString** () const  
*Returns a String object representing this **UUID** (p. 3969).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw ( lang::exceptions::UnsupportedOperationException )  
*The node value associated with this **UUID** (p. 3969).*
- virtual long long **timestamp** () throw ( lang::exceptions::UnsupportedOperationException )  
*The timestamp value associated with this **UUID** (p. 3969).*
- virtual int **clockSequence** () throw ( lang::exceptions::UnsupportedOperationException )  
*The clock sequence value associated with this **UUID** (p. 3969).*
- virtual int **variant** () throw ( lang::exceptions::UnsupportedOperationException )  
*The variant number associated with this **UUID** (p. 3969).*
- virtual int **version** () throw ( lang::exceptions::UnsupportedOperationException )  
*The version number associated with this **UUID** (p. 3969).*

## Static Public Member Functions

- static **UUID** **randomUUID** ()  
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3969).*
- static **UUID** **nameUUIDFromBytes** (const std::vector< char > &name)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3969) based on the specified byte array.*
- static **UUID** **nameUUIDFromBytes** (const char \*name, std::size\_t size)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3969) based on the specified byte array.*
- static **UUID** **fromString** (const std::string &name) throw (lang::exceptions::IllegalArgumentException )  
*Creates a **UUID** (p. 3969) from the string standard representation as described in the **toString()** (p. 3974) method.*

### 6.863.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 3969)). A **UUID** (p. 3969) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 3969) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 3969) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000 version
0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFF node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 3969). The bit layout described above is valid only for a **UUID** (p. 3969) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 3969). There are four different basic types of UUIDs: time-based, DCE **security** (p. 164), name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

### 6.863.2 Constructor & Destructor Documentation

#### 6.863.2.1 decaf::util::UUID::UUID (long long *mostSigBits*, long long *leastSigBits*)

Constructs a new **UUID** (p. 3969) using the specified data. *mostSigBits* is used for the most significant 64 bits of the **UUID** (p. 3969) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 3969).

**Parameters:***mostSigBits**leastSigBits***6.863.2.2** virtual decaf::util::UUID::~~UUID () [virtual]**6.863.3 Member Function Documentation****6.863.3.1** virtual int decaf::util::UUID::clockSequence () throw (  
lang::exceptions::UnsupportedOperationException ) [virtual]

The clock sequence value associated with this **UUID** (p.3969). The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p.3969). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p.3969).

The clockSequence value is only meaningful in a time-based **UUID** (p.3969), which has version type 1. If this **UUID** (p.3969) is not a time-based **UUID** (p.3969) then this method throws UnsupportedOperationException.

**Returns:**

the clockSequeunce associated with a V1 **UUID** (p.3969)

**Exceptions:***UnsupportedOperationException***6.863.3.2** virtual int decaf::util::UUID::compareTo (const UUID & *value*) const  
[virtual]

Compare the given **UUID** (p.3969) to this one.

**Parameters:**

*value* - the **UUID** (p.3969) to compare to

**6.863.3.3** virtual bool decaf::util::UUID::equals (const UUID & *value*) const  
[virtual]

Compares this **UUID** (p.3969) to the one given, returns true if they are equal.

**Parameters:**

*value* - the **UUID** (p.3969) to compare to.

**Returns:**

true if UUIDs are the same.

**6.863.3.4**    `static UUID decaf::util::UUID::fromString (const std::string & name)  
                  throw ( lang::exceptions::IllegalArgumentException )    [static]`

Creates a **UUID** (p. 3969) from the string standard representation as described in the **toString()** (p. 3974) method.

**Parameters:**

*name* - a string to be used to construct a **UUID** (p. 3969).

**Returns:**

type 3 **UUID** (p. 3969)

**6.863.3.5**    `virtual long long decaf::util::UUID::getLeastSignificantBits () const  
                  [virtual]`

**Returns:**

the most significant 64 bits of this UUID's 128 bit value.

**6.863.3.6**    `virtual long long decaf::util::UUID::getMostSignificantBits () const  
                  [virtual]`

**Returns:**

the most significant 64 bits of this UUID's 128 bit value.

**6.863.3.7**    `static UUID decaf::util::UUID::nameUUIDFromBytes (const char *  
                  name, std::size_t size)    [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 3969) based on the specified byte array.

**Parameters:**

*name* - a byte array to be used to construct a **UUID** (p. 3969).

*size* - the size of the byte array, or number of bytes to use.

**Returns:**

type 3 **UUID** (p. 3969)

**6.863.3.8**    `static UUID decaf::util::UUID::nameUUIDFromBytes (const std::vector<  
                  char > & name)    [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 3969) based on the specified byte array.

**Parameters:**

*name* - a byte array to be used to construct a **UUID** (p. 3969).

**Returns:**

type 3 **UUID** (p. 3969)

**6.863.3.9** `virtual long long decaf::util::UUID::node () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The node value associated with this **UUID** (p. 3969). The 48 bit node value is constructed from the node field of this **UUID** (p. 3969). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 3969) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 3969), which has version type 1. If this **UUID** (p. 3969) is not a time-based **UUID** (p. 3969) then this method throws `UnsupportedOperationException`.

**Returns:**

the node value of this **UUID** (p. 3969)

**Exceptions:**

*`UnsupportedOperationException`*

**6.863.3.10** `virtual bool decaf::util::UUID::operator< (const UUID & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.863.3.11** `virtual bool decaf::util::UUID::operator== (const UUID & value) const [virtual]`

Compares equality between this object and the one passed.

**Parameters:**

*value* - the value to be compared to this one.

**Returns:**

true if this object is equal to the one passed.

**6.863.3.12** `static UUID decaf::util::UUID::randomUUID () [static]`

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3969). The **UUID** (p. 3969) is generated using a cryptographically strong pseudo random number generator.

**Returns:**

type 4 **UUID** (p. 3969)

**6.863.3.13** `virtual long long decaf::util::UUID::timestamp () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The timestamp value associated with this **UUID** (p. 3969). The 60 bit timestamp value is constructed from the `time_low`, `time_mid`, and `time_hi` fields of this **UUID** (p. 3969). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 3969), which has version type 1. If this **UUID** (p. 3969) is not a time-based **UUID** (p. 3969) then this method throws `UnsupportedOperationException`.

**Returns:**

the timestamp associated with a V1 **UUID** (p. 3969)

**Exceptions:**

*UnsupportedOperationException*

**6.863.3.14** `virtual std::string decaf::util::UUID::toString () const [virtual]`

Returns a `String` object representing this **UUID** (p. 3969). **UUID**'s are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

**Returns:**

formatted string for this **UUID** (p. 3969)

**6.863.3.15** `virtual int decaf::util::UUID::variant () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The variant number associated with this **UUID** (p. 3969). The variant number describes the layout of the **UUID** (p. 3969). The variant number has the following meaning:

\* 0 Reserved for NCS backward compatibility \* 2 The Leach-Salz variant (used by this class) \* 6 Reserved, Microsoft Corporation backward compatibility \* 7 Reserved for future definition

**Returns:**

the variant associated with a V1 **UUID** (p. 3969)

**Exceptions:**

*UnsupportedOperationException*

**6.863.3.16** `virtual int decaf::util::UUID::version () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The version number associated with this **UUID** (p. 3969). The version number describes how this **UUID** (p. 3969) was generated. The version number has the following meaning:

\* 1 Time-based **UUID** (p. 3969) \* 2 DCE security (p. 164) **UUID** (p. 3969) \* 3 Name-based **UUID** (p. 3969) \* 4 Randomly generated **UUID** (p. 3969)

**Returns:**

the version associated with a V1 **UUID** (p. 3969)

**Exceptions:**

*UnsupportedOperationException*

The documentation for this class was generated from the following file:

- `src/main/decaf/util/UUID.h`



## 6.864 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal **commands** (p.87) into and out of packets or into and out of streams, Channels and Datagrams.

#include <src/main/activemq/wireformat/WireFormat.h> Inheritance diagram for activemq::wireformat::WireFormat:

### Public Member Functions

- virtual **~WireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out)=0  
throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in)=0  
throw ( **decaf::io::IOException** )  
*Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*
- virtual void **setVersion** (int version)=0  
*Set the Version.*
- virtual int **getVersion** () const =0  
*Get the Version.*
- virtual bool **hasNegotiator** () const =0  
*Returns true if this **WireFormat** (p.3976) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual bool **inReceive** () const =0  
*Indicates if the WireFormat object is in the process of receiving a message.*
- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw ( **decaf::lang::exceptions::UnsupportedOperationException** )  
*If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.*

### 6.864.1 Detailed Description

Provides a mechanism to marshal **commands** (p.87) into and out of packets or into and out of streams, Channels and Datagrams.

**Version:**

Revision 1.1

## 6.864.2 Constructor & Destructor Documentation

**6.864.2.1** `virtual activemq::wireformat::WireFormat::~~WireFormat () [inline, virtual]`

## 6.864.3 Member Function Documentation

**6.864.3.1** `virtual Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & transport) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

### Parameters:

*transport* (p. 97) - the Transport to Wrap the Negotiator around.

### Returns:

new instance of a **WireFormatNegotiator** (p. 4016) as a **Pointer<Transport>** (p. 2946).

### Exceptions:

*UnsupportedOperationException* if the **WireFormat** (p. 3976) doesn't have a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2890), and **activemq::wireformat::stomp::StompWireFormat** (p. 3644).

**6.864.3.2** `virtual int activemq::wireformat::WireFormat::getVersion () const [pure virtual]`

Get the Version.

### Returns:

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2892), and **activemq::wireformat::stomp::StompWireFormat** (p. 3644).

**6.864.3.3** `virtual bool activemq::wireformat::WireFormat::hasNegotiator () const [pure virtual]`

Returns true if this **WireFormat** (p. 3976) has a Negotiator that needs to wrap the Transport that uses it.

### Returns:

true if the **WireFormat** (p. 3976) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2892), and **activemq::wireformat::stomp::StompWireFormat** (p. 3644).

#### 6.864.3.4 virtual bool activemq::wireformat::WireFormat::inReceive () const [pure virtual]

Indicates if the WireFormat object is in the process of receiving a message. This is useful for monitoring inactivity and the **WireFormat** (p. 3976) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 3976) instance to determine if its busy or not and not mark the connection as inactive if so.

##### Returns:

true if the **WireFormat** (p. 3976) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2892), and **activemq::wireformat::stomp::StompWireFormat** (p. 3645).

#### 6.864.3.5 virtual void activemq::wireformat::WireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport \* *transport*, decaf::io::DataOutputStream \* *out*) throw ( decaf::io::IOException ) [pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

##### Parameters:

*command* The Command to Marshal

*transport* (p. 97) The Transport that called this method.

*out* The output stream to write the command to.

##### Exceptions:

*IOException*

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2894), and **activemq::wireformat::stomp::StompWireFormat** (p. 3645).

#### 6.864.3.6 virtual void activemq::wireformat::WireFormat::setVersion (int *version*) [pure virtual]

Set the Version.

##### Parameters:

*version* the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2896), and **activemq::wireformat::stomp::StompWireFormat** (p. 3645).

**6.864.3.7** `virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal (const activemq::transport::Transport * transport, decaf::io::DataInputStream * in) throw ( decaf::io::IOException )` [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

**Parameters:**

*transport* (p. 97) - Pointer to the **transport** (p. 97) that is making this request.

*in* - the input stream to read the command from.

**Returns:**

the newly marshaled Command, caller owns the pointer

**Exceptions:**

*IOException*

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2898), and **activemq::wireformat::stomp::StompWireFormat** (p. 3645).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormat.h`

## 6.865 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 3980) is the interface that all **WireFormatFactory** (p. 3980) classes must extend.

#include <src/main/activemq/wireformat/WireFormatFactory.h> Inheritance diagram for activemq::wireformat::WireFormatFactory:

### Public Member Functions

- virtual **~WireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties)=0 throw ( **decaf::lang::exceptions::IllegalStateException** )  
*Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.865.1 Detailed Description

The **WireFormatFactory** (p. 3980) is the interface that all **WireFormatFactory** (p. 3980) classes must extend. The Factory creates a **WireFormat** (p. 3976) Object based on the properties that are set in the passed **Properties** object.

### 6.865.2 Constructor & Destructor Documentation

**6.865.2.1** virtual **activemq::wireformat::WireFormatFactory::~~WireFormatFactory** () [inline, virtual]

### 6.865.3 Member Function Documentation

**6.865.3.1** virtual **Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw ( **decaf::lang::exceptions::IllegalStateException** ) [pure virtual]

Creates a new **WireFormat** (p. 3976) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters:

*properties* - the **Properties** for this **WireFormat** (p. 3976)

#### Returns:

Pointer to a new instance of a **WireFormat** (p. 3976) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2899), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 3647).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatFactory.h`

## 6.866 activemq::commands::WireFormatInfo Class Reference

#include <src/main/activemq/commands/WireFormatInfo.h> Inheritance diagram for activemq::commands::WireFormatInfo:

### Public Member Functions

- **WireFormatInfo** ()
- virtual **~WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataStructure** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const  
*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( **exceptions::ActiveMQException** )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- int **getVersion** () const  
*Get the current Wireformat Version.*
- void **setVersion** (int version)  
*Set the current Wireformat Version.*
- long long **getMaxInactivityDuration** () const  
*Returns the currently configured Max Inactivity duration.*
- void **setMaxInactivityDuration** (long long maxInactivityDuration)  
*Sets the Max inactivity duration value.*

- **long long getMaxInactivityDurationInitialDelay () const**  
*Returns the currently configured Max Inactivity Initial Delay duration.*
- **void setMaxInactivityDurationInitialDelay (long long maxInactivityDurationInitialDelay)**  
*Sets the Max inactivity initial delay duration value.*
- **bool isStackTraceEnabled () const**  
*Checks if the stackTraceEnabled flag is on.*
- **void setStackTraceEnabled (bool stackTraceEnabled)**  
*Sets if the stackTraceEnabled flag is on.*
- **bool isTcpNoDelayEnabled () const**  
*Checks if the tcpNoDelayEnabled flag is on.*
- **void setTcpNoDelayEnabled (bool tcpNoDelayEnabled)**  
*Sets if the tcpNoDelayEnabled flag is on.*
- **bool isCacheEnabled () const**  
*Checks if the cacheEnabled flag is on.*
- **void setCacheEnabled (bool cacheEnabled)**  
*Sets if the cacheEnabled flag is on.*
- **int getCacheSize () const**  
*Gets the Cache Size setting.*
- **void setCacheSize (int value)**  
*Sets the Cache Size setting.*
- **bool isTightEncodingEnabled () const**  
*Checks if the tightEncodingEnabled flag is on.*
- **void setTightEncodingEnabled (bool tightEncodingEnabled)**  
*Sets if the tightEncodingEnabled flag is on.*
- **bool isSizePrefixDisabled () const**  
*Checks if the sizePrefixDisabled flag is on.*
- **void setSizePrefixDisabled (bool sizePrefixDisabled)**  
*Sets if the sizePrefixDisabled flag is on.*
- **const std::vector< unsigned char > &getMagic () const**  
*Get the Magic field.*
- **void setMagic (const std::vector< unsigned char > &magic)**  
*Sets the value of the magic field.*



- `const std::vector< unsigned char > & getMarshalledProperties () const`  
*Get the marshalledProperties field.*
- `void setMarshalledProperties (const std::vector< unsigned char > &marshalledProperties)`  
*Sets the value of the marshalledProperties field.*
- `virtual const util::PrimitiveMap & getProperties () const`  
*Gets the Properties for this **Command** (p. 1194).*
- `virtual util::PrimitiveMap & getProperties ()`  
*Gets the Properties for this **Command** (p. 1194).*
- `virtual void setProperties (const util::PrimitiveMap &map)`  
*Sets the Properties for this **Command** (p. 1194).*
- `bool isValid () const`  
*Determines if we think this is a Valid **WireFormatInfo** (p. 3982) command.*
- `virtual bool isWireFormatInfo () const`
- `virtual void beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_ - UNUSED) throw ( decaf::io::IOException )`  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.*
- `virtual void afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_ - UNUSED) throw ( decaf::io::IOException )`  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*

## Static Public Attributes

- `static const unsigned char ID_ WIREFORMATINFO = 1`

## 6.866.1 Constructor & Destructor Documentation

**6.866.1.1** `activemq::commands::WireFormatInfo::WireFormatInfo ()`

**6.866.1.2** `virtual activemq::commands::WireFormatInfo::~~WireFormatInfo ()`  
[virtual]

## 6.866.2 Member Function Documentation

**6.866.2.1** `virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_ UNUSED) throw ( decaf::io::IOException )` [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

### Parameters:

*wireFormat* - the **wireformat** (p. 105) object to control unmarshaling

Reimplemented from `activemq::commands::BaseDataStructure` (p. 831).

**6.866.2.2** `virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw ( decaf::io::IOException )` [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

**Parameters:**

*wireFormat* - the wire formatting controller

Reimplemented from `activemq::commands::BaseDataStructure` (p. 831).

**6.866.2.3** `virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns:**

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1660).

**6.866.2.4** `virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure *src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns:**

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.866.2.5** `virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure *value) const` [virtual]

Compares the `DataStructure` (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 759).

**6.866.2.6 int activemq::commands::WireFormatInfo::getCacheSize () const**

Gets the Cache Size setting.

**Returns:**

currently set cache size.

**6.866.2.7 virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const [virtual]**

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataSet** (p. 1660) type copy.

Implements **activemq::commands::DataSet** (p. 1662).

**6.866.2.8 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const [inline]**

Get the Magic field.

**Returns:**

const reference to a std::vector<char>

**6.866.2.9 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties () const [inline]**

Get the marshalledProperties field.

**Returns:**

const reference to a std::vector<char>

**6.866.2.10 long long activemq::commands::WireFormatInfo::getMaxInactivityDuration () const**

Returns the currently configured Max Inactivity duration.

**Returns:**

the set inactivity duration value.

**6.866.2.11** `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitialDelay () const`

Returns the currently configured Max Inactivity Initial Delay duration.

**Returns:**

the set inactivity duration initial delay value.

**6.866.2.12** `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () [inline, virtual]`

Gets the Properties for this **Command** (p. 1194).

**Returns:**

the Properties object for this **Command** (p. 1194).

**6.866.2.13** `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () const [inline, virtual]`

Gets the Properties for this **Command** (p. 1194).

**Returns:**

the Properties object for this **Command** (p. 1194).

**6.866.2.14** `int activemq::commands::WireFormatInfo::getVersion () const [inline]`

Get the current Wireformat Version.

**Returns:**

int that identifies the version

**6.866.2.15** `bool activemq::commands::WireFormatInfo::isCacheEnabled () const`

Checks if the cacheEnabled flag is on.

**Returns:**

true if the flag is on.

**6.866.2.16** `virtual bool activemq::commands::WireFormatInfo::isMarshalAware () const [inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

**Returns:**

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 832).

**6.866.2.17    bool activemq::commands::WireFormatInfo::isSizePrefixDisabled ()  
                  const**

Checks if the sizePrefixDisabled flag is on.

**Returns:**

true if the flag is on.

**6.866.2.18    bool activemq::commands::WireFormatInfo::isStackTraceEnabled ()  
                  const**

Checks if the stackTraceEnabled flag is on.

**Returns:**

true if the flag is on.

**6.866.2.19    bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled ()  
                  const**

Checks if the tcpNoDelayEnabled flag is on.

**Returns:**

true if the flag is on.

**6.866.2.20    bool activemq::commands::WireFormatInfo::isTightEncodingEnabled ()  
                  const**

Checks if the tightEncodingEnabled flag is on.

**Returns:**

true if the flag is on.

**6.866.2.21    bool activemq::commands::WireFormatInfo::isValid () const**

Determines if we think this is a Valid **WireFormatInfo** (p. 3982) command.

**Returns:**

true if its valid.

**6.866.2.22** `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo ()`  
`const [inline, virtual]`

**Returns:**

answers true to the isWireFormatInfo query

Reimplemented from `activemq::commands::BaseCommand` (p. 763).

**6.866.2.23** `void activemq::commands::WireFormatInfo::setCacheEnabled (bool`  
`cacheEnabled)`

Sets if the cacheEnabled flag is on.

**Parameters:**

*cacheEnabled* - true to turn flag is on

**6.866.2.24** `void activemq::commands::WireFormatInfo::setCacheSize (int value)`

Sets the Cache Size setting.

**Parameters:**

*value* - value to set to the cache size.

**6.866.2.25** `void activemq::commands::WireFormatInfo::setMagic (const`  
`std::vector< unsigned char > & magic) [inline]`

Sets the value of the magic field.

**Parameters:**

*magic* - const std::vector<char>

**6.866.2.26** `void activemq::commands::WireFormatInfo::setMarshaledProperties`  
`(const std::vector< unsigned char > & marshalledProperties) [inline]`

Sets the value of the marshalledProperties field.

**Parameters:**

*marshalledProperties* The Byte Array vector that contains the marshaled form of the Message (p. 2516) properties, this is the data sent over the wire.

**6.866.2.27** `void activemq::commands::WireFormatInfo::setMaxInactivityDuration`  
`(long long maxInactivityDuration)`

Sets the Max inactivity duration value.

**Parameters:**

*maxInactivityDuration* - max time a client can be inactive.

**6.866.2.28** void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitialDelay (long long *maxInactivityDurationInitialDelay*)

Sets the Max inactivity initial delay duration value.

**Parameters:**

*maxInactivityDurationInitialDelay* - time before the inactivity delay is checked.

**6.866.2.29** virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & *map*) [inline, virtual]

Sets the Properties for this **Command** (p. 1194).

**Parameters:**

*map* - PrimitiveMap to copy

**6.866.2.30** void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool *sizePrefixDisabled*)

Sets if the sizePrefixDisabled flag is on.

**Parameters:**

*sizePrefixDisabled* - true to turn flag is on

**6.866.2.31** void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool *stackTraceEnabled*)

Sets if the stackTraceEnabled flag is on.

**Parameters:**

*stackTraceEnabled* - ture to turn flag is on

**6.866.2.32** void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*)

Sets if the tcpNoDelayEnabled flag is on.

**Parameters:**

*tcpNoDelayEnabled* - ture to turn flag is on

**6.866.2.33** void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool *tightEncodingEnabled*)

Sets if the tightEncodingEnabled flag is on.

**Parameters:**

*tightEncodingEnabled* - true to turn flag is on

**6.866.2.34** `void activemq::commands::WireFormatInfo::setVersion (int version)`  
[inline]

Set the current Wireformat Version.

**Parameters:**

*version* - int that identifies the version

**6.866.2.35** `virtual std::string activemq::commands::WireFormatInfo::toString ()`  
`const` [virtual]

Returns a string containing the information for this **DataSet** (p.1660) such as its type and value of its elements.

**Returns:**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 763).

**6.866.2.36** `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::commands::WireFormatInfo::visit (ac-`  
`tivemq::state::CommandVisitor * visitor) throw (`  
`exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

**Returns:**

a **Response** (p. 3285) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1198).

## 6.866.3 Field Documentation

**6.866.3.1** `const unsigned char activemq::commands::WireFormatInfo::ID _-`  
`WIREFORMATINFO = 1` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/WireFormatInfo.h`



## 6.867 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3992).

#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.867.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3992).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.867.2 Constructor & Destructor Documentation

**6.867.2.1** `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.867.2.2** `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.867.3 Member Function Documentation

**6.867.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.867.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.867.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.867.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.867.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.867.3.6** virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.867.3.7** virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**

## 6.868 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3996).

#include <src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.868.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3996).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.868.2 Constructor & Destructor Documentation

**6.868.2.1** `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.868.2.2** `virtual activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.868.3 Member Function Documentation

**6.868.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.868.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.868.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1624).

**6.868.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1631).

**6.868.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the `marshal` (p. 107).

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1638).

**6.868.3.6** virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.868.3.7** virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**WireFormatInfoMarshaller.h**



## 6.869 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4000).

#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.869.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4000).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.869.2 Constructor & Destructor Documentation

**6.869.2.1** `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.869.2.2** `virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.869.3 Member Function Documentation

**6.869.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.869.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.869.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.869.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.869.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.869.3.6** virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.869.3.7** virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h**

## 6.870 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4004).

#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.870.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4004).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.870.2 Constructor & Destructor Documentation

**6.870.2.1** `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.870.2.2** `virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.870.3 Member Function Documentation

**6.870.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.870.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.870.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.870.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.870.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.870.3.6** virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.870.3.7** virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**WireFormatInfoMarshaller.h**



## 6.871 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4008).

#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.871.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4008).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.871.2 Constructor & Destructor Documentation

**6.871.2.1** `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.871.2.2** `virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.871.3 Member Function Documentation

**6.871.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.871.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.871.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.871.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.871.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.871.3.6** virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.871.3.7** virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**

## 6.872 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4012).

#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.872.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4012).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.872.2 Constructor & Destructor Documentation

**6.872.2.1** `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

**6.872.2.2** `virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

## 6.872.3 Member Function Documentation

**6.872.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.872.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.872.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1624).

**6.872.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1631).

**6.872.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs*) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters:**

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

**Returns:**

int value indicating the size of the marshaled object.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1638).

**6.872.3.6** virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p.107).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1645).

**6.872.3.7** virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1652).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**



## 6.873 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 4016) which allows a **WireFormat** (p. 3976) to.

#include <src/main/activemq/wireformat/WireFormatNegotiator.h>Inheritance diagram for activemq::wireformat::WireFormatNegotiator:

### Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)  
*Constructor.*
- virtual ~**WireFormatNegotiator** ()

#### 6.873.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 4016) which allows a **WireFormat** (p. 3976) to.

#### 6.873.2 Constructor & Destructor Documentation

##### 6.873.2.1 activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator (const **Pointer**< **transport::Transport** > & *next*) [inline]

Constructor.

##### Parameters:

*next* - the next **Transport** in the chain

##### 6.873.2.2 virtual activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

## 6.874 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 3976) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

### Public Member Functions

- virtual **~WireFormatRegistry** ()
- **WireFormatFactory** \* **findFactory** (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets a Registered **WireFormatFactory** (p. 3980) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **WireFormatFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )

*Registers a new **WireFormatFactory** (p. 3980) with this Registry.*

- void **unregisterFactory** (const std::string &name)

*Unregisters the Factory with the given name and deletes that instance of the Factory.*

- std::vector< std::string > **getWireFormatNames** () const

*Retrieves a list of the names of all the Registered WireFormat's in this Registry.*

### Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()

*Gets the single instance of the **WireFormatRegistry** (p. 4017).*

#### 6.874.1 Detailed Description

Registry of all **WireFormat** (p. 3976) Factories that are available to the client at runtime. New WireFormat's must have a factory registered here before a connection attempt is made.

Since:

3.0

## 6.874.2 Constructor & Destructor Documentation

**6.874.2.1** virtual  
activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry ()  
[virtual]

## 6.874.3 Member Function Documentation

**6.874.3.1** WireFormatFactory\* ac-  
tivemq::wireformat::WireFormatRegistry::findFactory (const std::string  
& *name*) const throw ( decaf::lang::exceptions::NoSuchElementException  
)

Gets a Registered **WireFormatFactory** (p. 3980) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

### Parameters:

*name* The name of the Factory to find in the Registry.

### Returns:

the Factory registered under the given name.

### Exceptions:

*NoSuchElementException* if no factory is registered with that name.

**6.874.3.2** static WireFormatRegistry& ac-  
tivemq::wireformat::WireFormatRegistry::getInstance ()  
[static]

Gets the single instance of the **WireFormatRegistry** (p. 4017).

### Returns:

reference to the single instance of this Registry

**6.874.3.3** std::vector<std::string> ac-  
tivemq::wireformat::WireFormatRegistry::getWireFormatNames ()  
const

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

### Returns:

stl vector of strings with all the **WireFormat** (p. 3976) names registered.

**6.874.3.4**   `void activemq::wireformat::WireFormatRegistry::registerFactory`  
                  `(const std::string & name, WireFormatFactory * factory)`  
                  `throw ( decaf::lang::exceptions::IllegalArgumentException,`  
                  `decaf::lang::exceptions::NullPointerException )`

Registers a new **WireFormatFactory** (p. 3980) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

**Parameters:**

*name*   The name of the new Factory to register.  
    *factory*   The new Factory to add to the Registry.

**Exceptions:**

*IllegalArgumentException*   if name is the empty string.  
    *NullPointerException*   if the Factory is Null.

**6.874.3.5**   `void activemq::wireformat::WireFormatRegistry::unregisterFactory`  
                  `(const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

**Parameters:**

*name*   Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatRegistry.h`

## 6.875 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {.

#include <src/main/activemq/transport/inactivity/WriteChecker.h>Inheritance diagram for activemq::transport::inactivity::WriteChecker:

### Public Member Functions

- **WriteChecker** (**InactivityMonitor** \*parent)
- virtual **~WriteChecker** ()
- virtual void **run** ()

*Run method - called by the Thread class in the context of the thread.*

### 6.875.1 Detailed Description

Runnable class used by the {.

See also:

**InactivityMonitor** (p. 2004)} to make periodic writes to the underlying **transport** (p. 97) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

Since:

3.1.0

### 6.875.2 Constructor & Destructor Documentation

**6.875.2.1** **activemq::transport::inactivity::WriteChecker::WriteChecker** (**InactivityMonitor** \* *parent*)

**6.875.2.2** **virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker** ()  
[virtual]

### 6.875.3 Member Function Documentation

**6.875.3.1** **virtual void activemq::transport::inactivity::WriteChecker::run** ()  
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3325).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**WriteChecker.h**

## 6.876 decaf::io::Writer Class Reference

#include <src/main/decaf/io/Writer.h> Inheritance diagram for decaf::io::Writer:

### Public Member Functions

- **Writer** ()
- virtual **~Writer** ()
- virtual void **write** (char v) throw ( decaf::io::IOException )  
*Writes an single byte char value.*
- virtual void **write** (const std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Writes an array of Chars.*
- virtual void **write** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const std::string &str) throw ( decaf::io::IOException )  
*Writes a string.*
- virtual void **write** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a string.*
- virtual **decaf::lang::Appendable** & **append** (char value) throw ( decaf::io::IOException )  
*Appends the specified character to this Appendable.*
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )  
*Appends the specified character sequence to this Appendable.*
- virtual **decaf::lang::Appendable** & **append** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Appends a subsequence of the specified character sequence to this Appendable.*

### Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Override this method to customize the functionality of the method `write( char* buffer, int size, int offset, int length )`.*

- virtual void **doWriteChar** (char v) throw ( decaf::io::IOException )
- virtual void **doWriteVector** (const std::vector< char > &buffer) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )
- virtual void **doWriteString** (const std::string &str) throw ( decaf::io::IOException )
- virtual void **doWriteStringBounded** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **decaf::lang::Appendable** & **doAppendChar** (char value) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable** & **doAppendCharSequence** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable** & **doAppendCharSequenceStartEnd** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.876.1 Constructor & Destructor Documentation

**6.876.1.1** decaf::io::Writer::Writer ()

**6.876.1.2** virtual decaf::io::Writer::~~Writer () [virtual]

## 6.876.2 Member Function Documentation

**6.876.2.1** virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence \* *csq*, int *start*, int *end*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Appends a subsequence of the specified character sequence to this Appendable.

### Parameters:

*csq* - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

*start* The index of the first character in the subsequence.

*end* The index of the character following the last character in the subsequence.

### Returns:

a Reference to this Appendable

### Exceptions:

**Exception** if an error occurs.

**IndexOutOfBoundsException** *start* is greater than *end*, or *end* is greater than *csq.length()*

Implements **decaf::lang::Appendable** (p. 726).

**6.876.2.2** `virtual decaf::lang::Appendable& decaf::io::Writer::append (const decaf::lang::CharSequence * csq) throw ( decaf::io::IOException )`  
[virtual]

Appends the specified character sequence to this Appendable.

**Parameters:**

*csq* The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

**Returns:**

a Reference to this Appendable.

**Exceptions:**

*Exception* if an error occurs.

Implements **decaf::lang::Appendable** (p. 726).

**6.876.2.3** `virtual decaf::lang::Appendable& decaf::io::Writer::append (char value)`  
`throw ( decaf::io::IOException )` [virtual]

Appends the specified character to this Appendable.

**Parameters:**

*value* The character to append.

**Returns:**

a Reference to this Appendable

**Exceptions:**

*Exception* if an error occurs.

Implements **decaf::lang::Appendable** (p. 726).



- 6.876.2.4 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendChar (char *value*) throw ( decaf::io::IOException ) [protected, virtual]
- 6.876.2.5 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequence (const decaf::lang::CharSequence \* *csq*) throw ( decaf::io::IOException ) [protected, virtual]
- 6.876.2.6 virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequenceStartEnd (const decaf::lang::CharSequence \* *csq*, int *start*, int *end*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]
- 6.876.2.7 virtual void decaf::io::Writer::doWriteArray (const char \* *buffer*, int *size*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]
- 6.876.2.8 virtual void decaf::io::Writer::doWriteArrayBounded (const char \* *buffer*, int *size*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, pure virtual]

Override this method to customize the functionality of the method `write( char* buffer, int size, int offset, int length )`. All subclasses must override this method to provide the basic **Writer** (p. 4021) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 2916).

- 6.876.2.9 virtual void decaf::io::Writer::doWriteChar (char *v*) throw ( decaf::io::IOException ) [protected, virtual]
- 6.876.2.10 virtual void decaf::io::Writer::doWriteString (const std::string & *str*) throw ( decaf::io::IOException ) [protected, virtual]
- 6.876.2.11 virtual void decaf::io::Writer::doWriteStringBounded (const std::string & *str*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]
- 6.876.2.12 virtual void decaf::io::Writer::doWriteVector (const std::vector< char > & *buffer*) throw ( decaf::io::IOException ) [protected, virtual]
- 6.876.2.13 virtual void decaf::io::Writer::write (const std::string & *str*, int *offset*, int *length*) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes a string.

#### Parameters:

*str* The string to be written.

*offset* The position in the array to start writing from.

*length* The number of bytes in the array to write.

**Exceptions:**

*IOException* (p. 2142) thrown if an error occurs.

*IndexOutOfBoundsException* if offset+length is greater than the string length.

**6.876.2.14** `virtual void decaf::io::Writer::write (const std::string & str) throw ( decaf::io::IOException )` [virtual]

Writes a string.

**Parameters:**

*str* The string to be written.

**Exceptions:**

*IOException* (p. 2142) thrown if an error occurs.

**6.876.2.15** `virtual void decaf::io::Writer::write (const char * buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes a byte array to the output stream.

**Parameters:**

*buffer* The byte array to write (cannot be NULL).

*size* The size in bytes of the buffer passed.

*offset* The position in the array to start writing from.

*length* The number of bytes in the array to write.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

*NullPointerException* if buffer is NULL.

*IndexOutOfBoundsException* if offset + length > size of the buffer.

**6.876.2.16** `virtual void decaf::io::Writer::write (const char * buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )` [virtual]

Writes a byte array to the output stream.

**Parameters:**

*buffer* The byte array to write (cannot be NULL).

*size* The size in bytes of the buffer passed.

**Exceptions:**

*IOException* (p. 2142) if an I/O error occurs.

*NullPointerException* if buffer is NULL.

**6.876.2.17** `virtual void decaf::io::Writer::write (const std::vector< char > & buffer)  
throw ( decaf::io::IOException ) [virtual]`

Writes an array of Chars.

**Parameters:**

*buffer* The array to be written.

**Exceptions:**

*IOException* (p. 2142) thrown if an error occurs.

**6.876.2.18** `virtual void decaf::io::Writer::write (char v) throw (  
decaf::io::IOException ) [virtual]`

Writes an single byte char value.

**Parameters:**

*v* The value to be written.

**Exceptions:**

*IOException* (p. 2142) thrown if an error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Writer.h`

## 6.877 decaf::security::auth::x500::X500Principal Class Reference

`#include <src/main/decaf/security/auth/x500/X500Principal.h>`Inheritance diagram for `decaf::security::auth::x500::X500Principal`:

### Public Member Functions

- virtual `~X500Principal ()`
- virtual `std::string getName () const =0`  
*Provides the name of this principal.*
- virtual void `getEncoded (std::vector< unsigned char > &output) const =0`
- virtual int `hashCode () const =0`

### 6.877.1 Constructor & Destructor Documentation

**6.877.1.1** virtual `decaf::security::auth::x500::X500Principal::~X500Principal ()`  
[inline, virtual]

### 6.877.2 Member Function Documentation

**6.877.2.1** virtual void `decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & output) const` [pure virtual]

**6.877.2.2** virtual `std::string decaf::security::auth::x500::X500Principal::getName () const` [pure virtual]

Provides the name of this principal.

#### Returns:

the name of this principal.

Implements `decaf::security::Principal` (p. 3026).

**6.877.2.3** virtual int `decaf::security::auth::x500::X500Principal::hashCode () const`  
[pure virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/auth/x500/X500Principal.h`

## 6.878 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

#include <src/main/decaf/security/cert/X509Certificate.h> Inheritance diagram for decaf::security::cert::X509Certificate:

### Public Member Functions

- virtual `~X509Certificate ()`
- virtual void `checkValidity ()` const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void `checkValidity (const decaf::util::Date &date)` const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int `getBasicConstraints ()` const =0
- virtual void `getIssuerUniqueID (std::vector< bool > &output)` const =0
- virtual const X500Principal \* `getIssuerX500Principal ()` const =0
- virtual void `getKeyUsage (std::vector< unsigned char > &output)` const =0
- virtual Date `getNotAfter ()` const =0
- virtual Date `getNotBefore ()` const =0
- virtual std::string `getSigAlgName ()` const =0
- virtual std::string `getSigAlgOID ()` const =0
- virtual void `getSigAlgParams (std::vector< unsigned char > &output)` const =0
- virtual void `getSignature (std::vector< unsigned char > &output)` const =0
- virtual void `getSubjectUniqueID (std::vector< bool > &output)` const =0
- virtual const X500Principal \* `getSubjectX500Principal ()` const =0
- virtual void `getTBSCertificate (std::vector< unsigned char > &output)` const =0 throw ( CertificateEncodingException )
- virtual int `getVersion ()` const =0

### 6.878.1 Detailed Description

Base interface for all identity certificates.

## 6.878.2 Constructor & Destructor Documentation

6.878.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ()  
[inline, virtual]

## 6.878.3 Member Function Documentation

6.878.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

6.878.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity () const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

6.878.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints () const [pure virtual]

6.878.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

6.878.3.5 virtual const X500Principal\* decaf::security::cert::X509Certificate::getIssuerX500Principal () const [pure virtual]

6.878.3.6 virtual void decaf::security::cert::X509Certificate::getKeyUsage (std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.7 virtual Date decaf::security::cert::X509Certificate::getNotAfter () const [pure virtual]

6.878.3.8 virtual Date decaf::security::cert::X509Certificate::getNotBefore () const [pure virtual]

6.878.3.9 virtual std::string decaf::security::cert::X509Certificate::getSigAlgName () const [pure virtual]

6.878.3.10 virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID () const [pure virtual]

6.878.3.11 virtual void decaf::security::cert::X509Certificate::getSigAlgParams (std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.12 virtual void decaf::security::cert::X509Certificate::getSignature (std::vector< unsigned char > & *output*) const [pure virtual]

6.878.3.13 virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID (std::vector< bool > & *output*) const [pure virtual]

6.878.3.14 virtual const X500Principal\* decaf::security::cert::X509Certificate::getSubjectX500Principal () const [pure virtual]

6.878.3.15 virtual void decaf::security::cert::X509Certificate::getTBSCertificate (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

6.878.3.16 virtual int decaf::security::cert::X509Certificate::getVersion () const [pure virtual]

- `src/main/decaf/security/cert/X509Certificate.h`

## 6.879 activemq::commands::XATransactionId Class Reference

#include <src/main/activemq/commands/XATransactionId.h> Inheritance diagram for activemq::commands::XATransactionId:

### Public Types

- typedef decaf::lang::PointerComparator< XATransactionId > COMPARATOR

### Public Member Functions

- XATransactionId ()
- XATransactionId (const XATransactionId &other)
- virtual ~XATransactionId ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual XATransactionId \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const DataStructure \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this DataStructure (p. 1660) such as its type and value of its elements.*
- virtual bool **equals** (const DataStructure \*value) const  
*Compares the DataStructure (p. 1660) passed in to this one, and returns if they are equivalent.*
- virtual int **getFormatId** () const
- virtual void **setFormatId** (int formatId)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &globalTransactionId)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const
- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &branchQualifier)
- virtual int **compareTo** (const XATransactionId &value) const
- virtual bool **equals** (const XATransactionId &value) const
- virtual bool **operator==** (const XATransactionId &value) const
- virtual bool **operator<** (const XATransactionId &value) const
- XATransactionId & **operator=** (const XATransactionId &other)



## Static Public Attributes

- static const unsigned char **ID\_XATRANSACTIONID** = 112

## Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

## 6.879.1 Member Typedef Documentation

- 6.879.1.1**    `typedef decaf::lang::PointerComparator<XATransactionId>  
activemq::commands::XATransactionId::COMPARATOR`

Reimplemented from `activemq::commands::TransactionId` (p. 3819).

## 6.879.2 Constructor & Destructor Documentation

- 6.879.2.1**    `activemq::commands::XATransactionId::XATransactionId ()`
- 6.879.2.2**    `activemq::commands::XATransactionId::XATransactionId (const  
XATransactionId & other)`
- 6.879.2.3**    `virtual activemq::commands::XATransactionId::~~XATransactionId ()  
[virtual]`

## 6.879.3 Member Function Documentation

- 6.879.3.1**    `virtual XATransactionId* ac-  
tivemq::commands::XATransactionId::cloneDataStructure  
() const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns:

new copy of this object.

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

- 6.879.3.2**    `virtual int activemq::commands::XATransactionId::compareTo (const  
XATransactionId & value) const [virtual]`

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

- 6.879.3.3**    `virtual void activemq::commands::XATransactionId::copyDataStructure  
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters:**

*src* - Source Object

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.879.3.4** `virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3820).

**6.879.3.5** `virtual bool activemq::commands::XATransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1660) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns:**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.879.3.6** `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ()` [virtual]

**6.879.3.7** `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const` [virtual]

**6.879.3.8** `virtual unsigned char activemq::commands::XATransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns:**

new **DataStructure** (p. 1660) type copy.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.879.3.9** `virtual int activemq::commands::XATransactionId::getFormatId () const` [virtual]

**6.879.3.10** `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ()` [virtual]

**6.879.3.11** `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId () const` [virtual]

**6.879.3.12** `virtual bool activemq::commands::XATransactionId::operator< (const XATransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.879.3.13** `XATransactionId& activemq::commands::XATransactionId::operator= (const XATransactionId & other)`

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.879.3.14** `virtual bool activemq::commands::XATransactionId::operator== (const XATransactionId & value) const` [virtual]

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

**6.879.3.15** `virtual void activemq::commands::XATransactionId::setBranchQualifier (const std::vector< unsigned char > & branchQualifier)` [virtual]

**6.879.3.16** `virtual void activemq::commands::XATransactionId::setFormatId (int formatId)` [virtual]

**6.879.3.17** `virtual void activemq::commands::XATransactionId::setGlobalTransactionId (const std::vector< unsigned char > & globalTransactionId)` [virtual]

**6.879.3.18** `virtual std::string activemq::commands::XATransactionId::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1660) such as its type and value of its elements.

#### Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::TransactionId` (p. 3821).

## 6.879.4 Field Documentation

- 6.879.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier` [protected]
- 6.879.4.2 `int activemq::commands::XATransactionId::formatId` [protected]
- 6.879.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId` [protected]
- 6.879.4.4 `const unsigned char activemq::commands::XATransactionId::ID_ - XATRANSACTIONID = 112` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

## 6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4036).

#include <src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.880.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4036).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.880.2 Constructor & Destructor Documentation

**6.880.2.1** `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.880.2.2** `virtual activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.880.3 Member Function Documentation

**6.880.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.880.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.880.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3844).

**6.880.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3844).

**6.880.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3845).

**6.880.3.6** virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3845).

**6.880.3.7** virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3846).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h



## 6.881 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4040).

#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.881.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4040).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.881.2 Constructor & Destructor Documentation

**6.881.2.1** `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.881.2.2** `virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.881.3 Member Function Documentation

**6.881.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.881.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.881.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3832).

**6.881.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3832).

**6.881.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3833).

**6.881.3.6** virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3833).

**6.881.3.7** virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3834).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h

## 6.882 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4044).

#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.882.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4044).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.882.2 Constructor & Destructor Documentation

**6.882.2.1** `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.882.2.2** `virtual activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.882.3 Member Function Documentation

**6.882.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.882.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.882.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3840).

**6.882.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3840).

**6.882.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3841).

**6.882.3.6** virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3841).

**6.882.3.7** virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3842).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h



## 6.883 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4048).

#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.883.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4048).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.883.2 Constructor & Destructor Documentation

**6.883.2.1** `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.883.2.2** `virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.883.3 Member Function Documentation

**6.883.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.883.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.883.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3828).

**6.883.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3828).

**6.883.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3829).

**6.883.3.6** virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3829).

**6.883.3.7** virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3830).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h

## 6.884 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class Reference

Marshaling **code** (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4052).

#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.884.1 Detailed Description

Marshaling **code** (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4052).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.884.2 Constructor & Destructor Documentation

**6.884.2.1** `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.884.2.2** `virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.884.3 Member Function Documentation

**6.884.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.884.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.884.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3836).

**6.884.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3836).

**6.884.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3837).

**6.884.3.6** virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3837).

**6.884.3.7** virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3838).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h



## 6.885 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class Reference

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4056).

#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.885.1 Detailed Description

Marshaling `code` (p. 1183) for Open Wire Format for **XATransactionIdMarshaller** (p. 4056).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.885.2 Constructor & Destructor Documentation

**6.885.2.1** `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

**6.885.2.2** `virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

## 6.885.3 Member Function Documentation

**6.885.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

### Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1611).

**6.885.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1617).

**6.885.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataOut* - BinaryWriter that provides that data sink

### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3824).

**6.885.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*dataIn* - BinaryReader that provides that data source

#### Exceptions:

*IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3824).

**6.885.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters:

*wireFormat* - describes the wire format of the broker

*dataStructure* - Object to be marshaled

*bs* - BooleanStream stream used to pack bits from the wire.

#### Returns:

int value indicating the size of the marshaled object.

#### Exceptions:

*IOException* if an error occurs during the `marshal` (p. 107).

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3825).

**6.885.3.6** virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker  
*dataStructure* - Object to be marshaled  
*dataOut* - BinaryReader that provides that data sink  
*bs* - BooleanStream stream used to pack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the **marshal** (p. 107).

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3825).

**6.885.3.7** virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs*) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters:**

*wireFormat* - describes the wire format of the broker.  
*dataStructure* - Object to be un-marshaled.  
*dataIn* - BinaryReader that provides that data.  
*bs* - BooleanStream stream used to unpack bits from the wire.

**Exceptions:**

*IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3826).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h

## 6.886 decaf::util::logging::XMLFormatter Class Reference

Format a **LogRecord** (p. 2413) into a standard XML format.

#include <src/main/decaf/util/logging/XMLFormatter.h>Inheritance diagram for decaf::util::logging::XMLFormatter:

### Public Member Functions

- **XMLFormatter** ()
- virtual **~XMLFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const  
*Converts a **LogRecord** (p. 2413) into an XML string.*
- virtual std::string **getHead** (const **Handler** \*handler)  
*Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.*
- virtual std::string **getTail** (const **Handler** \*handler)  
*Returns the tail string for a set of log records formatted as XML strings.*

### 6.886.1 Detailed Description

Format a **LogRecord** (p. 2413) into a standard XML format. TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 4060) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 1978).

Since:

1.0

### 6.886.2 Constructor & Destructor Documentation

6.886.2.1 decaf::util::logging::XMLFormatter::XMLFormatter ()

6.886.2.2 virtual decaf::util::logging::XMLFormatter::~~XMLFormatter ()  
[virtual]

### 6.886.3 Member Function Documentation

6.886.3.1 virtual std::string decaf::util::logging::XMLFormatter::format (const **LogRecord** & record) const [virtual]

Converts a **LogRecord** (p. 2413) into an XML string.

Parameters:

*record* The log record to be formatted.

**Returns:**

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 1964).

**6.886.3.2 virtual std::string decaf::util::logging::XMLFormatter::getHead (const Handler \* *handler*) [virtual]**

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

**Parameters:**

*handler* The output handler, may be NULL.

**Returns:**

the header string for log records formatted as XML strings.

**6.886.3.3 virtual std::string decaf::util::logging::XMLFormatter::getTail (const Handler \* *handler*) [virtual]**

Returns the tail string for a set of log records formatted as XML strings.

**Parameters:**

*handler* The output handler, may be NULL.

**Returns:**

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/XMLFormatter.h`

## 6.887 z\_stream\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

### Data Fields

- `Bytef * next_in`
- `uInt avail_in`
- `uLong total_in`
- `Bytef * next_out`
- `uInt avail_out`
- `uLong total_out`
- `char * msg`
- `struct internal_state FAR * state`
- `alloc_func zalloc`
- `free_func zfree`
- `voidpf opaque`
- `int data_type`
- `uLong Adler`
- `uLong reserved`

### 6.887.1 Field Documentation

6.887.1.1 `uLong z_stream_s::Adler`

6.887.1.2 `uInt z_stream_s::avail_in`

6.887.1.3 `uInt z_stream_s::avail_out`

6.887.1.4 `int z_stream_s::data_type`

6.887.1.5 `char* z_stream_s::msg`

6.887.1.6 `Bytef* z_stream_s::next_in`

6.887.1.7 `Bytef* z_stream_s::next_out`

6.887.1.8 `voidpf z_stream_s::opaque`

6.887.1.9 `uLong z_stream_s::reserved`

6.887.1.10 `struct internal_state FAR* z_stream_s::state` [read]

6.887.1.11 `uLong z_stream_s::total_in`

6.887.1.12 `uLong z_stream_s::total_out`

6.887.1.13 `alloc_func z_stream_s::zalloc`

6.887.1.14 `free_func z_stream_s::zfree`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`



## 6.888 decaf::util::zip::ZipException Class Reference

#include <src/main/decaf/util/zip/ZipException.h> Inheritance diagram for decaf::util::zip::ZipException:

### Public Member Functions

- **ZipException** () throw ()  
*Default Constructor.*
- **ZipException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const ZipException &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ZipException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **ZipException** \* clone () const  
*Clones this exception.*
- virtual ~**ZipException** () throw ()

### 6.888.1 Constructor & Destructor Documentation

#### 6.888.1.1 decaf::util::zip::ZipException::ZipException () throw () [inline]

Default Constructor.

#### 6.888.1.2 decaf::util::zip::ZipException::ZipException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

#### Parameters:

*ex* the exception to copy

**6.888.1.3** `decaf::util::zip::ZipException::ZipException (const ZipException & ex) throw () [inline]`

Copy Constructor.

**Parameters:**

*ex* the exception to copy, which is an instance of this type

**6.888.1.4** `decaf::util::zip::ZipException::ZipException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*cause* The exception that was the cause for this one to be thrown.

*msg* The message to report

... list of primitives that are formatted into the message

**6.888.1.5** `decaf::util::zip::ZipException::ZipException (const std::exception * cause) throw () [inline]`

Constructor.

**Parameters:**

*cause* Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

**6.888.1.6** `decaf::util::zip::ZipException::ZipException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

**Parameters:**

*file* The file name where exception occurs

*lineNumber* The line number where the exception occurred.

*msg* The message to report

... list of primitives that are formatted into the message

**6.888.1.7**    `virtual decaf::util::zip::ZipException::~~ZipException () throw ()`  
                  [inline, virtual]

## 6.888.2 Member Function Documentation

**6.888.2.1**    `virtual ZipException* decaf::util::zip::ZipException::clone () const`  
                  [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns:

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2144).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/ZipException.h`



# Chapter 7

## File Documentation

### 7.1 src/main/activemq/cmsutil/CachedConsumer.h      File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

#### Data Structures

- class **activemq::cmsutil::CachedConsumer**  
*A cached message consumer contained within a pooled session.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::CachedProducer**  
*A cached message producer contained within a pooled session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **activemq::cmsutil::CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 1170) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1324) to operate on.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

### Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**  
*Extends the `CmsAccessor` (p. 1153) to add support for resolving destination names.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**



## 7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

### Data Structures

- class **activemq::cmsutil::CmsTemplate**  
*CmsTemplate* (p. 1170) simplifies performing synchronous CMS operations.
- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::DestinationResolver**  
*Resolves a CMS destination name to a **Destination**.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**  
*Manages maps of names to topics and queues for a single session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::MessageCreator**  
*Creates the user-defined message to be sent by the `CmsTemplate` (p. 1170).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/STLMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::PooledSession**  
*A pooled session object that wraps around a delegate session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::ProducerCallback**  
*Callback for sending a message to a CMS destination.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
#include <decaf/util/StlList.h>
```

### Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**  
*Manages the lifecycle of a set of CMS resources.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

### Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**



## 7.13 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::SessionCallback**  
*Callback for executing any number of operations on a provided CMS Session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.14 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::SessionPool**  
*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.15 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQBlobMessage`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.17 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

### Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.18 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQMapMessage`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.19 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

### Data Structures

- class **activemq::commands::ActiveMQMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.20 src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/core/ActiveMQConnection.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
```

### Data Structures

- class `activemq::commands::ActiveMQMessageTemplate< T >`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`



## 7.21 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQObjectMessage`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.22 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQQueue`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.23 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.24 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

### Data Structures

- class **activemq::commands::ActiveMQTempDestination**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::commands**

## 7.25 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQTempQueue`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.26 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQTempTopic`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.27 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQTextMessage`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.28 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::commands::ActiveMQTopic`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`



## 7.29 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/commands/Command.h>
```

### Data Structures

- class **activemq::commands::BaseCommand**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.30 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <string>
#include <sstream>
```

### Data Structures

- class **activemq::commands::BaseDataStructure**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::commands**

## 7.31 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::commands::BooleanExpression**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.32 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::BrokerError**  
*This class represents an Exception sent from the Broker.*
- struct **activemq::commands::BrokerError::StackTraceElement**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.33 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::BrokerId`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.34 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::BrokerInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.35 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class `activemq::commands::Command`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::state`
- namespace `activemq::commands`

## 7.36 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionControl**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.37 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionError**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.38 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.39 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ConnectionInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerControl**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.41 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.42 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.43 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ControlCommand`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.44 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::DataArrayResponse**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.45 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::DataResponse`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.46 src/main/activemq/commands/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

### Data Structures

- class **activemq::commands::DataStructure**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.47 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::DestinationInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.48 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::DiscoveryEvent**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.49 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ExceptionResponse`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.50 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::FlushCommand`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.51 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::IntegerResponse**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.52 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalQueueAck**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.53 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::JournalTopicAck`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.54 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTrace**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.55 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTransaction**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.56 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::KeepAliveInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.57 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::LastPartialCommand`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.58 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::LocalTransactionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.59 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataSet.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::Message**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**
- namespace **activemq::commands**

## 7.60 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **cms::Message**  
*Root of all messages.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.61 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageAck**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.62 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageDispatch**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.63 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::MessageDispatchNotification`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.64 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::MessageId`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.65 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::MessagePull`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.66 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::NetworkBridgeFilter`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.67 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::PartialCommand`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.68 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ProducerAck**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.69 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ProducerId`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.70 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ProducerInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.71 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataSet.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::RemoveInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.72 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.73 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ReplayCommand`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.74 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::Response**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.75 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SessionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SessionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.77 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::ShutdownInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.78 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SubscriptionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.79 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class `activemq::commands::TransactionId`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.80 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::TransactionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.81 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

### Data Structures

- class `activemq::commands::WireFormatInfo`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::commands`

## 7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::XATransactionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.83 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::core::ActiveMQAckHandler**  
*Interface class that is used to give CMS Messages an interface to Ack themselves with.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**
- namespace **activemq::core**

## 7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::core::ActiveMQConnection**  
*Concrete connection used for all connectors to the ActiveMQ broker.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**



## 7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

### Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

### Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 273) class.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::core::ActiveMQConstants**  
*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.*
- class **activemq::core::ActiveMQConstants::StaticInitializer**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/core/RedeliveryPolicy.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

### Data Structures

- class **activemq::core::ActiveMQConsumer**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

### Data Structures

- class **activemq::core::ActiveMQProducer**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.90 src/main/activemq/core/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/MessageEnumeration.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

### Data Structures

- class **activemq::core::ActiveMQQueueBrowser**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.91 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::core::ActiveMQSession`

### Namespaces

- namespace `activemq`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `activemq::core`

## 7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::core::ActiveMQSessionExecutor**  
*Delegate dispatcher for a single session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**



## 7.93 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **activemq::core::ActiveMQTransactionContext**

*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.94 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::core::DispatchData**

*Simple POCO that contains the information necessary to route a message to a specified consumer.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.95 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::core::Dispatcher**  
*Interface for an object responsible for dispatching messages to consumers.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.96 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class `activemq::core::MessageDispatchChannel`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::core`

## 7.97 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/PrefetchPolicy.h>
```

### Data Structures

- class **activemq::core::policies::DefaultPrefetchPolicy**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.98 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/RedeliveryPolicy.h>
```

### Data Structures

- class **activemq::core::policies::DefaultRedeliveryPolicy**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.99 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::core::PrefetchPolicy**

*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.100 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::core::RedeliveryPolicy**

*Interface for a **RedeliveryPolicy** (p. 3177) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**



## 7.101 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/exceptions/ActiveMQException.h>
```

### Data Structures

- class **activemq::core::Synchronization**

*Transacted Object **Synchronization** (p. 3715), used to sync the events of a Transaction with the items in the Transaction.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.102 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

### Data Structures

- class `activemq::exceptions::ActiveMQException`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::exceptions`

## 7.103 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

### Data Structures

- class `activemq::exceptions::BrokerException`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::exceptions`

## 7.104 src/main/activemq/exceptions/ExceptionDefines.h File Reference

### Defines

- `#define AMQ_CATCH_RETHROW(type)`  
*Macro for catching and re-throwing an exception of a given type.*
- `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`  
*Macro for catching an exception of one type and then re-throwing as another type.*
- `#define AMQ_CATCHALL_THROW(type)`  
*A catch-all that throws a known exception.*
- `#define AMQ_CATCHALL_NOTHROW()`  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- `#define AMQ_CATCH_NOTHROW(type)`  
*Macro for catching and re-throwing an exception of a given type.*

### 7.104.1 Define Documentation

#### 7.104.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

##### Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

##### Parameters:

***sourceType*** the type of the exception to be caught.

***targetType*** the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

#### 7.104.1.2 `#define AMQ_CATCH_NOTHROW(type)`

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

**Parameters:**

*type* The type of the exception to throw (e.g. ActiveMQException ).

### 7.104.1.3 #define AMQ\_CATCH\_RETHROW(type)

**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

**Parameters:**

*type* The type of the exception to throw (e.g. ActiveMQException ).

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

### 7.104.1.4 #define AMQ\_CATCHALL\_NOTHROW()

**Value:**

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
    "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

### 7.104.1.5 #define AMQ\_CATCHALL\_THROW(type)

**Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
    "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

**Parameters:**

*type* the type of exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`,  
`activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`,  
and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

## 7.105 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference

### Defines

- `#define DECAF_CATCH_RETHROW(type)`  
*Macro for catching and rethrowing an exception of a given type.*
- `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`  
*Macro for catching an exception of one type and then rethrowing as another type.*
- `#define DECAF_CATCHALL_THROW(type)`  
*A catch-all that throws a known exception.*
- `#define DECAF_CATCHALL_NOTHROW()`  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- `#define DECAF_CATCH_NOTHROW(type)`  
*Macro for catching and rethrowing an exception of a given type.*

### 7.105.1 Define Documentation

#### 7.105.1.1 `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

##### Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

##### Parameters:

- sourceType*** the type of the exception to be caught.
- targetType*** the type of the exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

#### 7.105.1.2 `#define DECAF_CATCH_NOTHROW(type)`

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

**Parameters:**

*type* The type of the exception to throw (e.g. Exception ).

**7.105.1.3 #define DECAF\_CATCH\_RETHROW(type)****Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

**Parameters:**

*type* The type of the exception to throw (e.g. Exception ).

Referenced by `decaf::util::PriorityQueue< E >::add()`.

**7.105.1.4 #define DECAF\_CATCHALL\_NOTHROW()****Value:**

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

**7.105.1.5 #define DECAF\_CATCHALL\_THROW(type)****Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

**Parameters:**

*type* the type of exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.



## 7.106 src/main/activemq/io/LoggingInputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **activemq::io::LoggingInputStream**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.107 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

### Data Structures

- class **activemq::io::LoggingOutputStream**  
*OutputStream filter that just logs the data being written.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.108 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

### Data Structures

- class `activemq::library::ActiveMQCPP`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::library`

## 7.109 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::state::CommandVisitor**

*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**
- namespace **activemq::state**

## 7.110 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

## Data Structures

- class `activemq::state::CommandVisitorAdapter`

*Default Implementation of a **CommandVisitor** (p. 1200) that returns NULL for all calls.*

## Namespaces

- namespace `activemq`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `activemq::state`

## 7.111 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::state::ConnectionState`

### Namespaces

- namespace `activemq`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `activemq::state`

## 7.112 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class `activemq::state::ConnectionStateTracker`

### Namespaces

- namespace `activemq`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `activemq::state`



## 7.113 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::state::ConsumerState`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::state`

## 7.114 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::state::ProducerState**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.115 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::state::SessionState`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::state`

## 7.116 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::state::Tracked**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.117 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

### Data Structures

- class `activemq::state::TransactionState`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::state`

## 7.118 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

### Data Structures

- class **activemq::threads::CompositeTask**  
*Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1223).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.119 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::threads::CompositeTaskRunner**

A *Task* (p. 3734) Runner that can contain one or more *CompositeTasks* that are each checked for pending work and run if any is present in the order that the tasks were added.

### Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

## 7.120 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::threads::DedicatedTaskRunner**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**



## 7.121 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::threads::Task**

*Represents a unit of work that requires one or more iterations to complete.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.122 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/threads/Task.h>
```

### Data Structures

- class **activemq::threads::TaskRunner**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.123 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::transport::AbstractTransportFactory**  
*Abstract implementation of the **TransportFactory** (p. 3889) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3889) instances.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.124 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

### Data Structures

- class **activemq::transport::CompositeTransport**

*A Composite **Transport** (p. 3883) is a **Transport** (p. 3883) implementation that is composed of several **Transports**.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.125 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

### Data Structures

- class **activemq::transport::correlator::FutureResponse**

*A container that holds a response object.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

## 7.126 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

### Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

*This type of **transport** (p. 97) filter is responsible for correlating asynchronous responses with requests.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

## 7.127 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::DefaultTransportListener**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.128 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

### Data Structures

- class `activemq::transport::failover::BackupTransport`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::transport`
- namespace `activemq::transport::failover`



## 7.129 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

### Data Structures

- class `activemq::transport::failover::BackupTransportPool`

### Namespaces

- namespace `activemq`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `activemq::transport`
- namespace `activemq::transport::failover`

## 7.130 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class `activemq::transport::failover::CloseTransportsTask`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::transport`
- namespace `activemq::transport::failover`

## 7.131 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransport**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.132 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**  
*Creates an instance of a **FailoverTransport** (p. 1873).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.133 src/main/activemq/transport/failover/FailoverTransportListener. File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransportListener**  
*Utility class used by the **Transport** (p. 3883) to perform the work of responding to events from the active **Transport** (p. 3883).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.134 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

### Data Structures

- class **activemq::transport::failover::URIPool**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.135 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

### Data Structures

- class **activemq::transport::inactivity::InactivityMonitor**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.136 src/main/activemq/transport/inactivity/ReadChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

### Data Structures

- class **activemq::transport::inactivity::ReadChecker**  
*Runnable class that is used by the {}.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**



## 7.137 src/main/activemq/transport/inactivity/WriteChecker.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/TimerTask.h>
```

### Data Structures

- class **activemq::transport::inactivity::WriteChecker**

*Runnable class used by the {.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.138 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

### Data Structures

- class **activemq::transport::IOTransport**  
*Implementation of the **Transport** (p. 3883) interface that performs marshaling of **commands** (p. 87) to IO streams.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::logging::LoggingTransport**  
*A **transport** (p. 97) filter that logs **commands** (p. 87) as they are sent/received.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::logging**

## 7.140 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

### Data Structures

- class **activemq::transport::mock::InternalCommandListener**  
*Listens for Commands sent from the **MockTransport** (p. 2768).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.141 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

### Data Structures

- class **activemq::transport::mock::MockTransport**

*The **MockTransport** (p. 2768) defines a base level **Transport** (p. 3883) class that is intended to be used in place of an a regular protocol **Transport** (p. 3883) such as TCP.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.142 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

### Data Structures

- class **activemq::transport::mock::MockTransportFactory**  
*Manufactures MockTransports, which are objects that read from input streams and write to output streams.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.143 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

### Data Structures

- class **activemq::transport::mock::ResponseBuilder**  
*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.144 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransport.h>
```

### Data Structures

- class **activemq::transport::tcp::SslTransport**  
*Transport (p. 3883) for connecting to a Broker using an SSL Socket.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**



## 7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransportFactory.h>
```

### Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.146 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

### Data Structures

- class **activemq::transport::tcp::TcpTransport**

*Implements a TCP/IP based **transport** (p. 97) filter, this **transport** (p. 97) is meant to wrap an instance of an **IOTransport** (p. 2145).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.147 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

### Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**  
*Factory Responsible for creating the **TcpTransport** (p. 3752).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.148 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

### Data Structures

- class **activemq::transport::Transport**  
*Interface for a **transport** (p. 97) layer for command objects.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::transport**

## 7.149 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::TransportFactory**  
*Defines the interface for Factories that create Transports or TransportFilters.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.150 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

### Data Structures

- class **activemq::transport::TransportFilter**

*A filter on the **transport** (p. 97) layer.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.151 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::TransportListener**  
*A listener of asynchronous **exceptions** (p. 92) from a command **transport** (p. 97) object.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.152 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::transport::TransportRegistry**  
*Registry of all **Transport** (p. 3883) Factories that are available to the client at runtime.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**



## 7.153 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::util::ActiveMQProperties**

*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3126) object.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.154 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>
```

### Data Structures

- class **activemq::util::CMSExceptionSupport**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

### Defines

- **#define AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION()**  
*Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.*

#### 7.154.1 Define Documentation

##### 7.154.1.1 #define AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION()

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need

to throw an exception of MessageNotReadableException for instance.

```

Referenced by    activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::acknowledge(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::clearBody(),   activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::clearProperties(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::equals(),      activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getBooleanProperty(),      activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getByteProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getCMSMessageID(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getIntProperty(),   activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getLongProperty(),  activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getPropertyNames(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getShortProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::getStringProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::propertyExists(),   activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setByteProperty(),  activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSReplyTo(),    activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setIntProperty(),    activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setLongProperty(),   activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setShortProperty(),  and activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setStringProperty().

```

## 7.155 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>
```

### Data Structures

- class **activemq::util::CompositeData**  
*Represents a Composite URI.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.156 src/main/activemq/util/Config.h File Reference

### Defines

- `#define AMQCPP_API`

#### 7.156.1 Define Documentation

##### 7.156.1.1 `#define AMQCPP_API`

## 7.157 src/main/cms/Config.h File Reference

### Defines

- `#define CMS_API`

### 7.157.1 Define Documentation

#### 7.157.1.1 `#define CMS_API`

## 7.158 src/main/decaf/util/Config.h File Reference

### Defines

- `#define DECAF_API`
- `#define DECAF_UNUSED`

### 7.158.1 Define Documentation

7.158.1.1 `#define DECAF_API`

7.158.1.2 `#define DECAF_UNUSED`

## 7.159 src/main/activemq/util/IdGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <string>
```

### Data Structures

- class **activemq::util::IdGenerator**
- class **activemq::util::IdGenerator::StaticData**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**



## 7.160 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **activemq::util::LongSequenceGenerator**

*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.161 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <string>
```

### Data Structures

- class **activemq::util::MarshallingSupport**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.162 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **activemq::util::MemoryUsage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.163 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

### Data Structures

- class **activemq::util::PrimitiveList**  
*List of primitives.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.164 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

### Data Structures

- class **activemq::util::PrimitiveMap**  
*Map of named primitives.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.165 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

### Data Structures

- class **activemq::util::PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3012) from one type to another.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.166 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

### Data Structures

- class **activemq::util::PrimitiveValueNode**  
*Class that wraps around a single value of one of the many types.*
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**  
*Define a union type comprised of the various types.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.167 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::util::URISupport**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**



## 7.168 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::util::Usage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.169 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::MarshalAware**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

7.170

src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

File Reference

7.170 src/main/activemq/wireformat/openwire/marshal/BaseDataStream<sup>4245</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

```
#include <activemq/wireformat/openwire/utils/HexTable.h>
```

```
#include <activemq/commands/MessageId.h>
```

```
#include <activemq/commands/ProducerId.h>
```

```
#include <activemq/commands/TransactionId.h>
```

```
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**

*Base class for all Marshallers that **marshal** (p. 107) DataStructures to and from the wire using the OpenWire protocol.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.171 src/main/activemq/wireformat/openwire/marshal/DataStreamM File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq:wireformat::openwire::marshal::DataStreamMarshaller**

*Base class for all classes that **marshal** (p. 107) **commands** (p. 87) for Openwire.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**
- namespace **activemq:wireformat::openwire::marshal**

7.172

src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File

Reference

7.172 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

*This class wraps the functionality needed to **marshal** (p. 107) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.173 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 213).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.174 src/main/activemq/wireformat/openwire/mar-  
shal/v2/ActiveMQBlobMessageMarshaller.h File

Reference

~~7.174~~ src/main/activemq/wireformat/openwire/marsh<sup>4249</sup>al/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *ActiveMQBlobMessageMarshaller* (p. 221).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.175 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 209).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.176 src/main/activemq/wireformat/openwire/mar-  
shal/v4/ActiveMQBlobMessageMarshaller.h File

Reference

7.176 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4251</sup>~~al/v4/ActiveMQ~~  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format for ActiveMQBlobMessageMarshaller* (p. 217).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.177 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 225).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.178 src/main/activemq/wireformat/openwire/mar-  
shal/v6/ActiveMQBlobMessageMarshaller.h File

Reference

7.178 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4253</sup>~~al/v6/ActiveMQ~~

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for **ActiveMQBlobMessageMarshaller**  
(p. 229).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.179 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 253).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.180 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQBytesMessageMarshaller.h File

Reference

7.180 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4255</sup>/v2/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 269).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 249).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.182 src/main/activemq/wireformat/openwire/mar-  
shal/v4/ActiveMQBytesMessageMarshaller.h File

Reference

7.182 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4257</sup>/v4/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *ActiveMQBytesMessageMarshaller*  
(p. 257).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.183 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 261).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.184 src/main/activemq/wireformat/openwire/mar-  
shal/v6/ActiveMQBytesMessageMarshaller.h File

Reference

~~7.184~~ <sup>4259</sup> src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 265).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.185 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 337).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.186 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQDestinationMarshaller.h File

Reference

7.186 src/main/activemq/wireformat/openwire/marsh<sup>4261</sup>/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 349).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.187 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 333).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.188 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQDestinationMarshaller.h File

Reference

7.188 src/main/activemq/wireformat/openwire/marsh<sup>4263</sup>  
shal/v4/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 341).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.189 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 345).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.190 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQDestinationMarshaller.h File

Reference

~~7.190~~ src/main/activemq/wireformat/openwire/marsh<sup>4265</sup>shal/v6/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 353).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.191 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 378).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.192 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQMapMessageMarshaller.h File

Reference

7.192 ~~src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ~~<sup>4267</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 390).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.193 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 374).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.194 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQMapMessageMarshaller.h File

Reference

~~7.194~~ src/main/activemq/wireformat/openwire/marsh<sup>4269</sup>shal/v4/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 382).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.195 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 386).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.196 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQMapMessageMarshaller.h File

Reference

7.196 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4271</sup>/v6/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *ActiveMQMapMessageMarshaller* (p. 394).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.197 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 405).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.198 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQMessageMarshaller.h File

Reference

7.198 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4273</sup>/v2/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 417).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.199 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 401).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.200 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQMessageMarshaller.h File

Reference

~~7.200~~ src/main/activemq/wireformat/openwire/marsh<sup>4275</sup>/v4/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 409).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.201 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 413).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.202 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQMessageMarshaller.h File

Reference

7.202 ~~src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQ~~<sup>4277</sup>  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQMessageMarshaller (p. 421).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.203 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 450).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.204 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQObjectMessageMarshaller.h File

Reference

~~7.204~~ src/main/activemq/wireformat/openwire/marsh<sup>4279</sup>al/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshall/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.205 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 446).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.206 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQObjectMessageMarshaller.h File

Reference

7.206 <sup>4281</sup>src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQO

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 454).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.207 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 458).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.208 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQObjectMessageMarshaller.h File

Reference

7.208 <sup>4283</sup>src/main/activemq/wireformat/openwire/marsh/v6/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 466).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.209 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 495).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.210

src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h

File Reference

7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h<sup>4285</sup>

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 507).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.211 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.212

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h<sup>4287</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.213 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 503).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.214

src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h

File Reference

7.214 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h<sup>4289</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQQueueMarshaller** (p. 511).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.215 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 556).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.216 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQStreamMessageMarshaller.h File

Reference

7.216 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4291</sup>/v2/ActiveMQS

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 568).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.217 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 552).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.218 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQStreamMessageMarshaller.h File

Reference

7.218 <sup>4293</sup>src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQS  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 560).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.219 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 564).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.220 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQStreamMessageMarshaller.h File

Reference

7.220 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4295</sup>/v6/ActiveMQS  
File Reference

```
#include <activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 572).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.221 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 584).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.222 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQTempDestinationMarshaller.h File

Reference

7.222 <sup>4297</sup>src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 596).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.223 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 580).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.224 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQTempDestinationMarshaller.h File

Reference

7.224 <sup>4299</sup>src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 588).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.225 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 592).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.226 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQTempDestinationMarshaller.h File

Reference

7.226 <sup>4301</sup>src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 600).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.227 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 613).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.228 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQTempQueueMarshaller.h File

Reference

7.228 <sup>4303</sup>src/main/activemq/wireformat/openwire/marsh/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 625).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.229 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 609).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.230 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQTempQueueMarshaller.h File

Reference

7.230 <sup>4305</sup>src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 617).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.231 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 621).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.232 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQTempQueueMarshaller.h File

Reference

7.232 <sup>4307</sup>src/main/activemq/wireformat/openwire/marsh/v6/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 629).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 646).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.234 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQTempTopicMarshaller.h File

Reference

7.234 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4309</sup>/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.235 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 638).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.236 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQTempTopicMarshaller.h File

Reference

7.236 <sup>4311</sup>src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 642).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.237 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 650).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.238 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQTempTopicMarshaller.h File

Reference

7.238 <sup>4313</sup>src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.239 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 675).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.240 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQTextMessageMarshaller.h File

Reference

~~7.240~~ src/main/activemq/wireformat/openwire/marsh<sup>4315</sup>  
shal/v2/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 687).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.241 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 667).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.242 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ActiveMQTextMessageMarshaller.h File

Reference

7.242 <sup>4317</sup>src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 671).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.243 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ActiveMQTextMessageMarshaller (p. 679).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.244 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ActiveMQTextMessageMarshaller.h File

Reference

~~7.244~~ src/main/activemq/wireformat/openwire/marsh<sup>4319</sup>/v6/ActiveMQ

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 683).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.245 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 703).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.246

src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h

File Reference

7.246 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ<sup>4321</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 715).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.247 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 695).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.248

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h

File Reference

7.248 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ<sup>4323</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 699).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.249 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 707).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.250

src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h

File Reference

7.250 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQ<sup>4325</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ActiveMQTopicMarshaller** (p. 711).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.251 src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 779).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.252

src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h

File Reference

7.252 src/main/activemq/wireformat/openwire/marshal/v2/BaseComm<sup>4327</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 800).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.253 src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 765).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.254

src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h

File Reference

7.254 src/main/activemq/wireformat/openwire/marshal/v4/BaseComm<sup>4329</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 772).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.255 src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 786).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.256

src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h

File Reference

7.256 src/main/activemq/wireformat/openwire/marshal/v6/BaseComm<sup>4331</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BaseCommandMarshaller** (p. 793).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.257 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 880).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.258 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 892).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.259 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 872).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.260 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 876).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.261 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 884).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.262 src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerIdMarshaller** (p. 888).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.263 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 912).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



## 7.264 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 924).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.265 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 904).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.266 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 908).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.267 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 916).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.268 src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **BrokerInfoMarshaller** (p. 920).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1280).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.270 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ConnectionControlMarshaller.h File

Reference

7.270 <sup>4345</sup>src/main/activemq/wireformat/openwire/marshal/v2/Connection  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1292).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.271 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1272).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.272 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ConnectionControlMarshaller.h File

Reference

7.272 src/main/activemq/wireformat/openwire/marsh<sup>4347</sup>/v4/Connection  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1276).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1284).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.274 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ConnectionControlMarshaller.h File

Reference

7.274 <sup>4349</sup>src/main/activemq/wireformat/openwire/marshal/v6/Connection

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConnectionControlMarshaller (p. 1288).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.275 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1312).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.276

src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h

File Reference

7.276 <sup>4351</sup>src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1300).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.277 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1304).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.278

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h

File Reference

7.278 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h 4353  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1308).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.279 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1316).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.280

src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h

File Reference

7.280 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h 4355  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionErrorMarshaller** (p. 1320).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.281 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1343).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.282

src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

File Reference

7.282 src/main/activemq/wireformat/openwire/marshal/v2/Connection<sup>4357</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1331).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.283 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1335).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.284

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

File Reference

7.284 src/main/activemq/wireformat/openwire/marshal/v4/Connection<sup>4359</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1339).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.285 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1347).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.286

src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

File Reference

7.286 src/main/activemq/wireformat/openwire/marshal/v6/Connection<sup>4361</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionIdMarshaller** (p. 1351).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.287 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1373).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.288

src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h

File Reference

7.288 src/main/activemq/wireformat/openwire/marshal/v2/Connection<sup>4363</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1361).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.289 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1365).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.290

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h

File Reference

7.290 src/main/activemq/wireformat/openwire/marshal/v4/Connection<sup>4365</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1369).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.291 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1377).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.292

src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h

File Reference

7.292 src/main/activemq/wireformat/openwire/marshal/v6/Connection<sup>4367</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConnectionInfoMarshaller** (p. 1381).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.293 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1418).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.294 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ConsumerControlMarshaller.h File

Reference

7.294 <sup>4369</sup>src/main/activemq/wireformat/openwire/marsh/v2/ConsumerC

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConsumerControlMarshaller (p. 1406).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.295 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConsumerControlMarshaller (p. 1410).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.296 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ConsumerControlMarshaller.h File

Reference

7.296 <sup>4371</sup>src/main/activemq/wireformat/openwire/marsh/v4/ConsumerC  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConsumerControlMarshaller (p. 1414).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.297 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ConsumerControlMarshaller** (p. 1422).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.298 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ConsumerControlMarshaller.h File

Reference

7.298 <sup>4373</sup>src/main/activemq/wireformat/openwire/marshal/v6/ConsumerC  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ConsumerControlMarshaller (p. 1426).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.299 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1447).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.300

src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File

Reference

7.300 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h 4375

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1435).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.301 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1439).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.302

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File

Reference

7.302 <sup>4377</sup>src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1443).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.303 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1451).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.304

src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h File

Reference

7.304 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h 4379

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerIdMarshaller** (p. 1455).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.305 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1480).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.306

src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

File Reference

~~7.306~~ src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h<sup>4381</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1468).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.307 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1472).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.308

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

File Reference

7.308 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h<sup>4383</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1476).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.309 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1484).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.310

src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

File Reference

~~7.310~~ src/main/activemq/wireformat/openwire/marshal/v6/ConsumerI<sup>4385</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ConsumerInfoMarshaller** (p. 1488).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.311 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1509).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.312 src/main/activemq/wireformat/openwire/mar-  
shal/v2/ControlCommandMarshaller.h File

Reference

7.312 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4387</sup>/v2/ControlCo  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1497).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.313 src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1501).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.314 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ControlCommandMarshaller.h File

Reference

7.314 <sup>4389</sup>src/main/activemq/wireformat/openwire/marshal/v4/ControlCo

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for ControlCommandMarshaller (p. 1505).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.315 src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1513).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.316 src/main/activemq/wireformat/openwire/mar-  
shal/v6/ControlCommandMarshaller.h File

Reference

7.316 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4391</sup>/v6/ControlCo  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **ControlCommandMarshaller** (p. 1517).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.317 src/main/activemq/wireformat/openwire/marshal/v1/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1543).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.318 src/main/activemq/wireformat/openwire/marsh  
shal/v2/DataArrayResponseMarshaller.h File

Reference

7.318 <sup>4393</sup>src/main/activemq/wireformat/openwire/marsh/v2/DataArray  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1531).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.319 src/main/activemq/wireformat/openwire/marshal/v3/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1535).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.320 src/main/activemq/wireformat/openwire/marsh  
shal/v4/DataArrayResponseMarshaller.h File

Reference

7.320 <sup>4395</sup>src/main/activemq/wireformat/openwire/marsh/v4/DataArray  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1539).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.321 src/main/activemq/wireformat/openwire/marshal/v5/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1547).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.322 src/main/activemq/wireformat/openwire/marsh  
shal/v6/DataArrayResponseMarshaller.h File

Reference

7.322 src/main/activemq/wireformat/openwire/marsh<sup>4397</sup>/v6/DataArray  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **DataArrayResponseMarshaller** (p. 1551).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.323 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1606).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.324

src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h

File Reference

7.324 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h<sup>4399</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1594).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.325 src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1598).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.326

src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h

File Reference

7.326 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h<sup>4401</sup>

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1602).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.327 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1586).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.328

src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h

File Reference

7.328 src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h<sup>4403</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DataResponseMarshaller** (p. 1590).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.329 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1744).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.330

src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h

File Reference

## 7.330 src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1732).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.331 src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1736).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.332

src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h

File Reference

## 7.332 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1740).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.333 src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1752).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.334

src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h

File Reference

7.334 src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h 4409

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DestinationInfoMarshaller** (p. 1748).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.335 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1778).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.336

src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h

File Reference

~~7.336~~ src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryE<sup>4411</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1766).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.337 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1770).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.338

src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h

File Reference

7.338 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryE<sup>4413</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1774).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.339 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1782).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.340

src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h

File Reference

7.340 src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h<sup>4415</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **DiscoveryEventMarshaller** (p. 1762).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.341 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1862).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.342 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ExceptionResponseMarshaller.h File

Reference

7.342 src/main/activemq/wireformat/openwire/marsh<sup>4417</sup>/v2/ExceptionR

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1846).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.343 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1850).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.344 src/main/activemq/wireformat/openwire/marsh  
shal/v4/ExceptionResponseMarshaller.h File

Reference

7.344 src/main/activemq/wireformat/openwire/marsh<sup>4419</sup>/v4/ExceptionR

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1858).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.345 src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1854).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.346 src/main/activemq/wireformat/openwire/marsh  
shal/v6/ExceptionResponseMarshaller.h File

Reference

7.346 src/main/activemq/wireformat/openwire/marsh<sup>4421</sup>/v6/ExceptionR

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ExceptionResponseMarshaller** (p. 1842).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.347 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1956).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.348

src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h

File Reference

7.348 src/main/activemq/wireformat/openwire/marshal/v2/FlushCom<sup>4423</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1944).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.349 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1948).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.350

src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h

File Reference

7.350 src/main/activemq/wireformat/openwire/marshal/v4/FlushCom<sup>4425</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1952).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.351 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1960).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.352

src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h

File Reference

7.352 src/main/activemq/wireformat/openwire/marshal/v6/FlushCom<sup>4427</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **FlushCommandMarshaller** (p. 1940).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.353 src/main/activemq/wireformat/openwire/marshal/v1/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**  
*Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2110).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.354

src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

File Reference

7.354 <sup>4429</sup> src/main/activemq/wireformat/openwire/marshal/v2/IntegerRes

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2098).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.355 src/main/activemq/wireformat/openwire/marshal/v3/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**  
*Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2102).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.356

src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h

File Reference

7.356 <sup>4431</sup>src/main/activemq/wireformat/openwire/marshal/v4/IntegerRes

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2106).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.357 src/main/activemq/wireformat/openwire/marshal/v5/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**  
*Marshaling `code` (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2114).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.358

src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h

File Reference

7.358 src/main/activemq/wireformat/openwire/marshal/v6/IntegerRes<sup>4433</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **IntegerResponseMarshaller** (p. 2094).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.359 src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2179).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.360 src/main/activemq/wireformat/openwire/marsh  
shal/v2/JournalQueueAckMarshaller.h File

Reference

7.360 <sup>4435</sup>src/main/activemq/wireformat/openwire/marsh/v2/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for JournalQueueAckMarshaller (p. 2163).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.361 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2171).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.362 src/main/activemq/wireformat/openwire/marsh  
shal/v4/JournalQueueAckMarshaller.h File

Reference

7.362 <sup>4437</sup>src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for JournalQueueAckMarshaller (p. 2175).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.363 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2167).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.364 src/main/activemq/wireformat/openwire/marsh  
shal/v6/JournalQueueAckMarshaller.h File

Reference

7.364 <sup>4439</sup>src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalQueueAckMarshaller** (p. 2159).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.365 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2208).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.366 src/main/activemq/wireformat/openwire/marsh-  
shal/v2/JournalTopicAckMarshaller.h File

Reference

7.366 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4441</sup>~~shal/v2/JournalTop~~  
File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2192).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.367 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2196).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.368 src/main/activemq/wireformat/openwire/marsh  
shal/v4/JournalTopicAckMarshaller.h File

Reference

7.368 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4443</sup>/v4/JournalTopic  
File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2204).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.369 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2188).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.370 src/main/activemq/wireformat/openwire/marsh  
shal/v6/JournalTopicAckMarshaller.h File

Reference

7.370 <sup>4445</sup>src/main/activemq/wireformat/openwire/marshal/v6/JournalTopic  
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTopicAckMarshaller** (p. 2200).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.371 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2231).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.372

src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h

File Reference

7.372 src/main/activemq/wireformat/openwire/marshal/v2/JournalTra<sup>4447</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2215).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.373 src/main/activemq/wireformat/openwire/marshal/v3/JournalTrace File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2219).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.374

src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

File Reference

7.374 src/main/activemq/wireformat/openwire/marshal/v4/JournalTra<sup>4449</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2227).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-*  
*ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.375 src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2235).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.376

src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h

File Reference

7.376 src/main/activemq/wireformat/openwire/marshal/v6/JournalTra<sup>4451</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **JournalTraceMarshaller** (p. 2223).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.377 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2262).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.378 src/main/activemq/wireformat/openwire/marsh-  
shal/v2/JournalTransactionMarshaller.h File

Reference

7.378 <sup>4453</sup>src/main/activemq/wireformat/openwire/marshal/v2/JournalTra  
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2246).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.379 src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2250).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.380 src/main/activemq/wireformat/openwire/marsh  
shal/v4/JournalTransactionMarshaller.h File

Reference

7.380 <sup>4455</sup>src/main/activemq/wireformat/openwire/marshal/v4/JournalTra  
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2258).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.381 src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2254).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.382 src/main/activemq/wireformat/openwire/marsh  
shal/v6/JournalTransactionMarshaller.h File

Reference

7.382 src/main/activemq/wireformat/openwire/marsh<sup>4457</sup>/v6/JournalTra

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **JournalTransactionMarshaller** (p. 2242).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.383 src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2289).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.384

src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h

File Reference

7.384 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h<sup>4459</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2273).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.385 src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2277).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.386

src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h

File Reference

7.386 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h<sup>4461</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2281).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.387 src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaler.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2285).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.388

src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

File Reference

~~7.388~~ <sup>4463</sup> src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2269).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.389 src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2324).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.390 src/main/activemq/wireformat/openwire/marsh  
shal/v2/LastPartialCommandMarshaller.h File

Reference

7.390 <sup>4465</sup>src/main/activemq/wireformat/openwire/marshal/v2/LastPartial

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2312).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.391 src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2308).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.392 src/main/activemq/wireformat/openwire/marsh  
shal/v4/LastPartialCommandMarshaller.h File

Reference

7.392 src/main/activemq/wireformat/openwire/marshal<sup>4467</sup>/v4/LastPartial

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2320).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.393 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2316).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.394 src/main/activemq/wireformat/openwire/marsh  
shal/v6/LastPartialCommandMarshaller.h File

Reference

7.394 <sup>4469</sup>src/main/activemq/wireformat/openwire/marsh/v6/LastPartial

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **LastPartialCommandMarshaller** (p. 2304).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.395 src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2371).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.396 src/main/activemq/wireformat/openwire/marsh  
shal/v2/LocalTransactionIdMarshaller.h File

Reference

7.396 src/main/activemq/wireformat/openwire/marsh<sup>4471</sup>/v2/LocalTrans

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2355).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.397 src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2359).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.398 src/main/activemq/wireformat/openwire/marsh  
shal/v4/LocalTransactionIdMarshaller.h File

Reference

7.398 <sup>4473</sup>src/main/activemq/wireformat/openwire/marshal/v4/LocalTrans

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code (p.1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p.2367).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.399 src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2363).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.400 src/main/activemq/wireformat/openwire/marsh  
shal/v6/LocalTransactionIdMarshaller.h File

Reference

7.400 <sup>4475</sup>src/main/activemq/wireformat/openwire/marshal/v6/LocalTrans

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2351).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.401 src/main/activemq/wireformat/openwire/marshal/v1/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.402 src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.403 src/main/activemq/wireformat/openwire/marshal/v3/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.404 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.405 src/main/activemq/wireformat/openwire/marshal/v5/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.406 src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.407 src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2581).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.408

src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h File

Reference

7.408 <sup>4483</sup>src/main/activemq/wireformat/openwire/marshal/v2/MessageAck

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2569).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.409 src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2573).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.410

src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h File

Reference

7.410<sup>4485</sup> src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2577).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.411 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2585).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.412

src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h File

Reference

7.412 <sup>4487</sup>src/main/activemq/wireformat/openwire/marshal/v6/MessageAck

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageAckMarshaller** (p. 2565).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.413 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2622).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.414 src/main/activemq/wireformat/openwire/mar-  
shal/v2/MessageDispatchMarshaller.h File

Reference

7.414 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4489</sup>/v2/MessageDi

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2606).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.415 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2610).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.416 src/main/activemq/wireformat/openwire/mar-  
shal/v4/MessageDispatchMarshaller.h File

Reference

7.416 src/main/activemq/wireformat/openwire/marshal/v4/MessageDi<sup>4491</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2618).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.417 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2614).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.418 src/main/activemq/wireformat/openwire/marsh  
shal/v6/MessageDispatchMarshaller.h File

Reference

7.418 src/main/activemq/wireformat/openwire/marsh<sup>4493</sup>/v6/MessageDi

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchMarshaller (p. 2626).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.419 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2651).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.420 src/main/activemq/wireformat/openwire/marsh  
shal/v2/MessageDispatchNotificationMarshaller.h File

Reference

7.420 <sup>4495</sup>src/main/activemq/wireformat/openwire/marshal/v2/MessageDi  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *MessageDispatchNotificationMarshaller* (p. 2639).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.421 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2643).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.422 src/main/activemq/wireformat/openwire/marsh  
shal/v4/MessageDispatchNotificationMarshaller.h File

Reference

7.422 src/main/activemq/wireformat/openwire/marsh<sup>4497</sup>/v4/MessageDi  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *MessageDispatchNotificationMarshaller* (p. 2647).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.423 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format for MessageDispatchNotificationMarshaller* (p. 2655).

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.424 src/main/activemq/wireformat/openwire/marsh  
shal/v6/MessageDispatchNotificationMarshaller.h File

Reference

7.424 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4499</sup>/v6/MessageDi  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for *MessageDispatchNotificationMarshaller* (p. 2635).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.425 src/main/activemq/wireformat/openwire/marshal/v1/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2688).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.426 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2668).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.427 src/main/activemq/wireformat/openwire/marshal/v3/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2680).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.428 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2672).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.429 src/main/activemq/wireformat/openwire/marshal/v5/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2676).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.430 src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageIdMarshaller** (p. 2684).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.431 src/main/activemq/wireformat/openwire/marshal/v1/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2713).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



## 7.432 src/main/activemq/wireformat/openwire/marshal/v2/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2703).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-*  
*ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.433 src/main/activemq/wireformat/openwire/marshal/v3/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2698).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.434 src/main/activemq/wireformat/openwire/marshal/v4/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2708).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.435 src/main/activemq/wireformat/openwire/marshal/v5/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2693).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.436 src/main/activemq/wireformat/openwire/marshal/v6/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessageMarshaller** (p. 2718).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.437 src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2760).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.438

src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

File Reference

7.438 src/main/activemq/wireformat/openwire/marshal/v2/MessagePu<sup>4513</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2744).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.439 src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2752).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.440

src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

File Reference

7.440 src/main/activemq/wireformat/openwire/marshal/v4/MessagePu<sup>4515</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2756).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.441 src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2748).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.442

src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h

File Reference

7.442 <sup>4517</sup>src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **MessagePullMarshaller** (p. 2764).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.443 src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2816).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.444 src/main/activemq/wireformat/openwire/marsh  
shal/v2/NetworkBridgeFilterMarshaller.h File

Reference

~~7.444~~ src/main/activemq/wireformat/openwire/marsh<sup>4519</sup>/v2/NetworkBr

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2796).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.445 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2808).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.446 src/main/activemq/wireformat/openwire/marsh  
shal/v4/NetworkBridgeFilterMarshaller.h File

Reference

7.446 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4521</sup>/v4/NetworkBr

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2812).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.447 src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2804).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.448 src/main/activemq/wireformat/openwire/marsh  
shal/v6/NetworkBridgeFilterMarshaller.h File

Reference

7.448 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4523</sup>/v6/NetworkBr

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

*Marshaling **code** (p. 1183) for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2800).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.449 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2942).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.450 src/main/activemq/wireformat/openwire/mar-  
shal/v2/PartialCommandMarshaller.h File

Reference

7.450 <sup>4525</sup>src/main/activemq/wireformat/openwire/marsh/v2/PartialCon  
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2926).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.451 src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2934).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.452 src/main/activemq/wireformat/openwire/marsh  
shal/v4/PartialCommandMarshaller.h File

Reference

7.452 <sup>4527</sup>src/main/activemq/wireformat/openwire/marshal/v4/PartialCon  
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2938).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.453 src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2930).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.454 src/main/activemq/wireformat/openwire/mar-  
shal/v6/PartialCommandMarshaller.h File

Reference

7.454 ~~src/main/activemq/wireformat/openwire/marsh~~<sup>4529</sup>/v6/PartialCon

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **PartialCommandMarshaller** (p. 2922).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.455 src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3060).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.456

src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h

File Reference

7.456 <sup>4531</sup>src/main/activemq/wireformat/openwire/marshal/v2/ProducerA

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3040).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.457 src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3048).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.458

src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h

File Reference

7.458 src/main/activemq/wireformat/openwire/marshal/v4/ProducerA<sup>4533</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3044).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.459 src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3052).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.460

src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h

File Reference

7.460 src/main/activemq/wireformat/openwire/marshal/v6/ProducerA<sup>4535</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerAckMarshaller** (p. 3056).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.461 src/main/activemq/wireformat/openwire/marshal/v1/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3092).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.462

src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File

Reference

7.462 <sup>4537</sup>src/main/activemq/wireformat/openwire/marshal/v2/ProducerId

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3072).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.463 src/main/activemq/wireformat/openwire/marshal/v3/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3080).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.464

src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h File

Reference

7.464 src/main/activemq/wireformat/openwire/marshal/v4/ProducerId<sup>4539</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3076).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.465 src/main/activemq/wireformat/openwire/marshal/v5/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3084).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.466

src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h File

Reference

7.466 src/main/activemq/wireformat/openwire/marshal/v6/ProducerId<sup>4541</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerIdMarshaller** (p. 3088).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.467 src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3109).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.468

src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

File Reference

7.468 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h<sup>4543</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3105).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.469 src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3117).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.470

src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h

File Reference

7.470 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h<sup>4545</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3101).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.471 src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3113).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.472

src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h

File Reference

7.472 src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h<sup>4547</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ProducerInfoMarshaller** (p. 3121).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.473 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3210).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.474

src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h File

Reference

7.474 <sup>4549</sup>src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfo

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3198).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.475 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3206).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.476

src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File

Reference

7.476 <sup>4551</sup>src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfo

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3218).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.477 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3214).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.478

src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h File

Reference

7.478 <sup>4553</sup>src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfo

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **RemoveInfoMarshaller** (p. 3202).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.479 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3227).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.480 src/main/activemq/wireformat/openwire/marsh  
shal/v2/RemoveSubscriptionInfoMarshaller.h File

Reference

7.480 <sup>4555</sup>src/main/activemq/wireformat/openwire/marsh/v2/RemoveSu

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for ***RemoveSubscriptionInfoMarshaller***  
(p. 3235).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.481 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3231).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.482 src/main/activemq/wireformat/openwire/mar-  
shal/v4/RemoveSubscriptionInfoMarshaller.h File

Reference

7.482 <sup>4557</sup>src/main/activemq/wireformat/openwire/marsh/v4/RemoveSu

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code* (p. 1183) for *Open Wire Format* for ***RemoveSubscriptionInfoMarshaller***  
(p. 3247).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.483 src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3243).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.484 src/main/activemq/wireformat/openwire/marsh  
shal/v6/RemoveSubscriptionInfoMarshaller.h File

Reference

~~7.484~~ src/main/activemq/wireformat/openwire/marsh<sup>4559</sup>al/v6/RemoveSubscriptionInfoMarshaller.h File

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3239).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.485 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3258).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.486 src/main/activemq/wireformat/openwire/mar-  
shal/v2/ReplayCommandMarshaller.h File

Reference

7.486 <sup>4561</sup>src/main/activemq/wireformat/openwire/marsh/v2/ReplayCon

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3262).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.487 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3266).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.488 src/main/activemq/wireformat/openwire/mar-  
shal/v4/ReplayCommandMarshaller.h File

Reference

7.488 <sup>4563</sup>src/main/activemq/wireformat/openwire/marsh/v4/ReplayCon

## File Reference

```
#include <activemq/wireformat/openwire/marshall/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3254).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.489 src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3274).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.490 src/main/activemq/wireformat/openwire/mar-  
shal/v6/ReplayCommandMarshaller.h File

Reference

~~7.490~~ <sup>4565</sup> src/main/activemq/wireformat/openwire/marsh/v6/ReplayCom  
File Reference

```
#include <activemq/wireformat/openwire/marshall/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for **ReplayCommandMarshaller** (p. 3270).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.491 src/main/activemq/wireformat/openwire/marshal/v1/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3315).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.492 src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3300).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.493 src/main/activemq/wireformat/openwire/marshal/v3/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3310).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.494 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3295).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.495 src/main/activemq/wireformat/openwire/marshal/v5/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3305).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



## 7.496 src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ResponseMarshaller** (p. 3320).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.497 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3403).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.498 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3383).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.499 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3399).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.500 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3387).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.501 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3395).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.502 src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionIdMarshaller** (p. 3391).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.503 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3419).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.504

src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h File

Reference

7.504 <sup>4579</sup>src/main/activemq/wireformat/openwire/marshal/v2/SessionInfo

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3427).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.505 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3423).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.506

src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h File

Reference

7.506 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfo<sup>4581</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3431).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.507 src/main/activemq/wireformat/openwire/marshal/v5/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3415).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.508

src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h File

Reference

7.508 src/main/activemq/wireformat/openwire/marshal/v6/SessionInfo<sup>4583</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SessionInfoMarshaller** (p. 3411).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.509 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3481).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.510

src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h

File Reference

7.510 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownI<sup>4585</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3477).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.511 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3489).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.512

src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

File Reference

7.512 src/main/activemq/wireformat/openwire/marshal/v4/ShutdownI<sup>4587</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3493).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.513 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3485).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.514

src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h

File Reference

7.514 src/main/activemq/wireformat/openwire/marshal/v6/ShutdownI<sup>4589</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **ShutdownInfoMarshaller** (p. 3473).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.515 src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3680).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.516

src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

File Reference

7.516 <sup>4591</sup>src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3696).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.517 src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3676).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.518

src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h

File Reference

7.518 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h<sup>4593</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3688).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.519 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3684).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.520

src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h

File Reference

7.520 src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h<sup>4595</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3692).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.521 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3827).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.522

src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h

File Reference

7.522 src/main/activemq/wireformat/openwire/marshal/v2/Transaction<sup>4597</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3831).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.523 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3835).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.524

src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h

File Reference

7.524 src/main/activemq/wireformat/openwire/marshal/v4/Transaction<sup>4599</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3839).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.525 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3823).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.526

src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h

File Reference

7.526 src/main/activemq/wireformat/openwire/marshal/v6/Transaction<sup>4601</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionIdMarshaller** (p. 3843).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.527 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3855).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.528

src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h

File Reference

7.528 src/main/activemq/wireformat/openwire/marshal/v2/Transaction<sup>4603</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3871).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.529 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3859).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.530

src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h

File Reference

7.530 src/main/activemq/wireformat/openwire/marshal/v4/Transaction<sup>4605</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3867).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.531 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3851).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.532

src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h

File Reference

7.532 src/main/activemq/wireformat/openwire/marshal/v6/Transaction<sup>4607</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **TransactionInfoMarshaller** (p. 3863).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.533 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4008).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.534

src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h

File Reference

7.534 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h 4609

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4000).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.535 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4012).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**



7.536

src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

File Reference

7.536 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 4004).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.537 src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3992).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.538

src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h

File Reference

7.538 src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h<sup>4613</sup>

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller**  
*Marshaling code (p. 1183) for Open Wire Format for **WireFormatInfoMarshaller** (p. 3996).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.539 src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4048).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.540 src/main/activemq/wireformat/openwire/marsh  
shal/v2/XATransactionIdMarshaller.h File

Reference

7.540 src/main/activemq/wireformat/openwire/marshal<sup>4615</sup>/v2/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4040).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.541 src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4052).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.542 src/main/activemq/wireformat/openwire/mar-  
shal/v4/XATransactionIdMarshaller.h File

Reference

7.542 <sup>4617</sup>src/main/activemq/wireformat/openwire/marsh/v4/XATransac  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4044).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.543 src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4056).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



7.544 src/main/activemq/wireformat/openwire/mar-  
shal/v6/XATransactionIdMarshaller.h File

Reference

7.544 <sup>4619</sup>src/main/activemq/wireformat/openwire/marsh/v6/XATransac

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller**

*Marshaling code (p. 1183) for Open Wire Format for XATransactionIdMarshaller (p. 4036).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.545 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.546 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.547 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq:wireformat::openwire::OpenWireFormatNegotiator**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**

## 7.548 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.549 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq:wireformat::openwire::utils::BooleanStream**  
*Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**
- namespace **activemq:wireformat::openwire::utils**

## 7.550 src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **activemq::wireformat::openwire::Utils::HexTable**

*The **HexTable** (p. 1984) class maps hexadecimal strings to the value of an index into the table, i.e.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::Utils**

## 7.551 src/main/activemq/wireformat/openwire/Utils/MessagePropertyInterceptor.h File Reference

```
#include <activemq/Util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/Util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **activemq::wireformat::openwire::Utils::MessagePropertyInterceptor**  
*Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::Utils**



## 7.552 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

### Data Structures

- class `activemq::wireformat::stomp::StompCommandConstants`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::stomp`

## 7.553 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompFrame**  
*A Stomp-level message frame that encloses all messages to and from the broker.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.554 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompHelper**  
*Utility Methods used when marshaling to and from StompFrame's.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.555 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.556 src/main/activemq/wireformat/stomp/StompWireFormatFactory File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**  
*Factory used to create the Stomp Wire Format instance.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.557 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormat**  
*Provides a mechanism to marshal **commands** (p. 87) into and out of packets or into and out of streams, Channels and Datagrams.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.558 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatFactory**

*The **WireFormatFactory** (p. 3980) is the interface that all **WireFormatFactory** (p. 3980) classes must extend.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.559 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatNegotiator**  
*Defines a **WireFormatNegotiator** (p. 4016) which allows a **WireFormat** (p. 3976) to.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**



## 7.560 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatRegistry**  
*Registry of all **WireFormat** (p. 3976) Factories that are available to the client at runtime.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.561 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

### Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 1056) object is used to send a message containing a stream of unsigned bytes.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.562 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Closeable**  
*Interface for a class that implements the close method.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.563 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::Closeable**  
*Interface for a class that implements the close method.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.564 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

### Data Structures

- class **cms::CMSException**

*CMS API Exception that is the base for all exceptions thrown from CMS classes.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.565 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

### Data Structures

- class **cms::CMSProperties**  
*Interface for a Java-like properties object.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.566 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::CMSSecurityException**

*This exception must be thrown when a provider rejects a user name/password submitted by a client.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.567 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stopable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

### Data Structures

- class **cms::Connection**  
*The client's connection to its provider.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.568 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

### Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1262) objects returned implement the **CMS Connection** (p. 1262) interface and hide the CMS Provider specific implementation details behind that interface.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.569 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 1385) object provides information describing the **Connection** (p. 1262) object.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.570 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

### Data Structures

- class **cms::DeliveryMode**

*This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.571 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

### Data Structures

- class **cms::Destination**

*A **Destination** (p. 1723) object encapsulates a provider-specific address.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.572 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1838) that is registered with the **Connection** (p. 1262).*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.573 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::IllegalStateException**

*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.574 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.575 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidClientIdException**

*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.576 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidDestinationException**

*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.577 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidSelectorException**

*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.578 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 2472) object is used to send a set of name-value pairs.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.579 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

### Data Structures

- class **cms::MessageConsumer**

*A client uses a `MessageConsumer` (p. 2589) to received messages from a destination.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.580 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageEnumeration**

*Defines an object that enumerates a collection of Messages.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.581 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3652) or **BytesMessage** (p. 1056) is being read.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.582 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.583 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

### Data Structures

- class **cms::MessageListener**

*A `MessageListener` (p. 2692) object is used to receive asynchronously delivered messages.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.584 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageNotReadableException**

*This exception must be thrown when a CMS client attempts to read a write-only message.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.585 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageNotWriteableException**

*This exception must be thrown when a CMS client attempts to write to a read-only message.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.586 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

### Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2725) object to send messages to a **Destination** (p. 1723).*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.587 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

### Data Structures

- class **cms::ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.588 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Queue**

*An interface encapsulating a provider-specific queue name.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.589 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::util::Queue< E >**

*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.590 src/main/cms/QueueBrowser.h File Reference

```
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageEnumeration.h>
```

### Data Structures

- class **cms::QueueBrowser**  
*This class implements in interface for browsing the messages in a **Queue** (p. 3148) without removing them.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.591 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Session**

*A **Session** (p. 3365) object is a single-threaded context for producing and consuming messages.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.592 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Startable**  
*Interface for a class that implements the start method.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.593 src/main/cms/Stopable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Stopable**  
*Interface for a class that implements the stop method.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.594 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

### Data Structures

- class **cms::StreamMessage**  
*Interface for a **StreamMessage** (p. 3652).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.595 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::TemporaryQueue**  
*Defines a Temporary **Queue** (p. 3148) based **Destination** (p. 1723).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.596 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::TemporaryTopic**  
*Defines a Temporary **Topic** (p. 3817) based **Destination** (p. 1723).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.597 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **cms::TextMessage**  
*Interface for a text message.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.598 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Topic**

*An interface encapsulating a provider-specific topic name.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.599 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::UnsupportedOperationException**

*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.600 src/main/decaf/lang/exceptions/UnsupportedOperationException File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.601 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <apr_pools.h>
```

### Data Structures

- class **decaf::internal::AprPool**

*Wraps an APR pool object so that classes in **decaf** (p. 143) can create a static member for use in static methods where apr function calls that need a pool are made.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**

## 7.602 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <apr_pools.h>
```

### Data Structures

- class **decaf::internal::DecafRuntime**  
*Handles APR initialization and termination.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**

## 7.603 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**  
*Wrapper Around the Standard error Output facility on the current platform.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.604 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardInputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.605 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardOutputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.606 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.607 src/main/decaf/internal/net/DefaultSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**



## 7.608 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/internal/util/Resource.h>
#include <decaf/internal/util/GenericResource.h>
```

### Data Structures

- class **decaf::internal::net::Network**

*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.609 src/main/decaf/internal/net/SocketFileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FileDescriptor.h>
```

### Data Structures

- class **decaf::internal::net::SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.610 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContext.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**  
*Default SSL Context manager for the Decaf library.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.611 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**  
*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.612 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**

*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.613 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**  
*Provides an SSLContext that wraps the OpenSSL API.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.614 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**  
*Container class for parameters that are Common to OpenSSL socket classes.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.615 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocket.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**  
*SSLServerSocket based on OpenSSL library code (p. 1183).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**



## 7.616 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**  
*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.617 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**

*Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.618 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**  
*Subclass of the standard `SocketException` that knows how to produce an error message from the `OpenSSL` error stack.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.619 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.620 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/io/InputStream.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**  
*An output stream for reading data from an OpenSSL Socket instance.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.621 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**  
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2858) instance.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.622 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
#include <decaf/io/IOException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocket**  
*Platform-independent implementation of the socket interface.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.623 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**

*Input stream for performing reads on a socket.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**



## 7.624 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**  
*Output stream for performing write operations on a socket.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.625 src/main/decaf/internal/net/URLEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

### Data Structures

- class **decaf::internal::net::URLEncoderDecoder**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.626 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

### Data Structures

- class **decaf::internal::net::URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.627 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::internal::net::URIType**  
*Basic type object that holds data that composes a given URI.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.628 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 163) package to create the various default version of the NIO interfaces.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.629 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

*This class defines six categories of operations upon byte buffers..*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.630 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.631 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**



## 7.632 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.633 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.634 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.635 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

### Data Structures

- class **decaf::internal::security::SecureRandomImpl**  
*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.637 src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

### Data Structures

- class **decaf::internal::security::SecureRandomImpl**  
*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.638 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

### Data Structures

- class **decaf::internal::util::ByteArrayAdapter**  
*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*
- union **decaf::internal::util::ByteArrayAdapter::Array**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**



## 7.640 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.641 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**

*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.642 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>  
#include <decaf/util/concurrent/TimeoutException.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**  
*Shared **internal** (p. 144) API for dual stacks and queues.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.643 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::TransferQueue< E >**

*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.644 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::TransferStack< E >**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.645 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.646 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.647 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::MutexHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**



## 7.648 src/main/decaf/internal/util/concurrent/windows/MutexHandle. File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::MutexHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-  
ments.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.649 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/internal/util/Resource.h>
```

### Data Structures

- class **decaf::internal::util::GenericResource**< **T** >  
*A Generic **Resource** (p. 3280) wraps some type and will delete it when the **Resource** (p. 3280) itself is deleted.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.650 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

### Data Structures

- class **decaf::internal::util::HexStringParser**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.651 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::internal::util::Resource**

*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.652 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::util::TimerTaskHeap**

*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.653 `src/main/decaf/internal/util/zip/crc32.h` File Reference

### Variables

- local const unsigned long FAR `crc_table` [TBLS][256]

### 7.653.1 Variable Documentation

7.653.1.1 local const unsigned long FAR `crc_table`[TBLS][256]

## 7.654 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

### Data Structures

- struct **ct\_data\_s**
- struct **tree\_desc\_s**
- struct **internal\_state**

### Defines

- #define **GZIP**
- #define **LENGTH\_CODES** 29
- #define **LITERALS** 256
- #define **L\_CODES** (LITERALS+1+LENGTH\_CODES)
- #define **D\_CODES** 30
- #define **BL\_CODES** 19
- #define **HEAP\_SIZE** (2\*L\_CODES+1)
- #define **MAX\_BITS** 15
- #define **INIT\_STATE** 42
- #define **EXTRA\_STATE** 69
- #define **NAME\_STATE** 73
- #define **COMMENT\_STATE** 91
- #define **HCRC\_STATE** 103
- #define **BUSY\_STATE** 113
- #define **FINISH\_STATE** 666
- #define **Freq** fc.freq
- #define **Code** fc.code
- #define **Dad** dl.dad
- #define **Len** dl.len
- #define **max\_insert\_length** max\_lazy\_match
- #define **put\_byte**(s, c) {s->pending\_buf[s->pending++] = (c);}
- #define **MIN\_LOOKAHEAD** (MAX\_MATCH+MIN\_MATCH+1)
- #define **MAX\_DIST**(s) ((s)->w\_size-MIN\_LOOKAHEAD)
- #define **WIN\_INIT** MAX\_MATCH
- #define **d\_code**(dist) ((dist) < 256 ? \_\_dist\_code[dist] : \_\_dist\_code[256+((dist)>>7)])
- #define **\_tr\_tally\_lit**(s, c, flush)
- #define **\_tr\_tally\_dist**(s, distance, length, flush)

### Typedefs

- typedef struct **ct\_data\_s** **ct\_data**
- typedef struct static\_tree\_desc\_s **static\_tree\_desc**
- typedef struct **tree\_desc\_s** **tree\_desc**
- typedef **ush** **Pos**
- typedef **Pos** FAR **Posf**
- typedef unsigned **IPos**
- typedef struct **internal\_state** **deflate\_state**

## Functions

- void ZLIB\_INTERNAL \_tr\_init **OF** ((deflate\_state \*s))
- int ZLIB\_INTERNAL \_tr\_tally **OF** ((deflate\_state \*s, unsigned dist, unsigned lc))
- void ZLIB\_INTERNAL \_tr\_flush\_block **OF** ((deflate\_state \*s, charf \*buf, ulg stored\_len, int last))

## Variables

- uch ZLIB\_INTERNAL \_length\_code []
- uch ZLIB\_INTERNAL \_dist\_code []

### 7.654.1 Define Documentation

#### 7.654.1.1 #define \_tr\_tally\_dist(s, distance, length, flush)

Value:

```
{ uch len = (length); \
    ush dist = (distance); \
    s->d_buf[s->last_lit] = dist; \
    s->l_buf[s->last_lit++] = len; \
    dist--; \
    s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
    s->dyn_dtree[d_code(dist)].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
}
```

#### 7.654.1.2 #define \_tr\_tally\_lit(s, c, flush)

Value:

```
{ uch cc = (c); \
    s->d_buf[s->last_lit] = 0; \
    s->l_buf[s->last_lit++] = cc; \
    s->dyn_ltree[cc].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
}
```



```

7.654.1.3  #define BL_CODES 19

7.654.1.4  #define BUSY_STATE 113

7.654.1.5  #define Code fc.code

7.654.1.6  #define COMMENT_STATE 91

7.654.1.7  #define d_code(dist) ((dist) < 256 ? _dist_code[dist] :
        _dist_code[256+((dist)>>7)])

7.654.1.8  #define D_CODES 30

7.654.1.9  #define Dad dl.dad

7.654.1.10 #define EXTRA_STATE 69

7.654.1.11 #define FINISH_STATE 666

7.654.1.12 #define Freq fc.freq

7.654.1.13 #define GZIP

7.654.1.14 #define HCRC_STATE 103

7.654.1.15 #define HEAP_SIZE (2*L_CODES+1)

7.654.1.16 #define INIT_STATE 42

7.654.1.17 #define L_CODES (LITERALS+1+LENGTH_CODES)

7.654.1.18 #define Len dl.len

7.654.1.19 #define LENGTH_CODES 29

7.654.1.20 #define LITERALS 256

7.654.1.21 #define MAX_BITS 15

7.654.1.22 #define MAX_DIST(s) ((s)->w_size-MIN_LOOKAHEAD)

7.654.1.23 #define max_insert_length max_lazy_match

7.654.1.24 #define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)

7.654.1.25 #define NAME_STATE 73

7.654.1.26 #define put_byte(s, c) {s->pending_buf[s->pending++] = (c);}

7.654.1.27 #define WIN_INIT MAX_MATCH

```

## 7.654.2 Typedef Documentation

```

7.654.2.1 typedef struct ct_data_s ct_data

```

---

```

7.654.2.2 typedef struct internal_state deflate_state

```

```

7.654.2.3 typedef unsigned IPos

```

```

7.654.2.4 typedef ush Pos

```

```

7.654.2.5 typedef unsigned short FAR Pos

```

## 7.655 src/main/decaf/internal/util/zip/gzguts.h File Reference

```
#include <stdio.h>
#include "zlib.h"
#include <fcntl.h>
```

### Data Structures

- struct **gz\_state**

### Defines

- #define **ZLIB\_INTERNAL**
- #define **local** static
- #define **zstrerror()** "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ\_NONE** 0
- #define **GZ\_READ** 7247
- #define **GZ\_WRITE** 31153
- #define **GZ\_APPEND** 1
- #define **LOOK** 0
- #define **COPY** 1
- #define **GZIP** 2
- #define **GT\_OFF(x)** (sizeof(int) == sizeof(z\_off64\_t) && (x) > gz\_intmax())

### Typedefs

- typedef **gz\_state FAR \* gz\_statep**

### Functions

- voidp malloc **OF** ((uInt size))
- void free **OF** ((voidpf ptr))
- ZEXTERN gzFile ZEXPORT gzopen64 **OF** ((const char \*, const char \*))
- ZEXTERN z\_off64\_t ZEXPORT gzseek64 **OF** ((gzFile, z\_off64\_t, int))
- ZEXTERN z\_off64\_t ZEXPORT gztell64 **OF** ((gzFile))
- void ZLIB\_INTERNAL gz\_error **OF** ((gz\_statep, int, const char \*))
- unsigned ZLIB\_INTERNAL gz\_intmax **OF** ((void))

## 7.655.1 Define Documentation

7.655.1.1 `#define COPY 1`

7.655.1.2 `#define GT_OFF(x) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())`

7.655.1.3 `#define GZ_APPEND 1`

7.655.1.4 `#define GZ_NONE 0`

7.655.1.5 `#define GZ_READ 7247`

7.655.1.6 `#define GZ_WRITE 31153`

7.655.1.7 `#define GZBUFSIZE 8192`

7.655.1.8 `#define GZIP 2`

7.655.1.9 `#define local static`

7.655.1.10 `#define LOOK 0`

7.655.1.11 `#define ZLIB_INTERNAL`

7.655.1.12 `#define zstrerror() "stdio error (consult errno)"`

## 7.655.2 Typedef Documentation

7.655.2.1 `typedef gz_state FAR* gz_statep`

## 7.655.3 Function Documentation

7.655.3.1 `unsigned ZLIB_INTERNAL gz_intmax OF ((void))`

7.655.3.2 `void ZLIB_INTERNAL gz_error OF ((gz_statep, int, const char *))`

7.655.3.3 `ZEXTERN z_off64_t ZEXPORT gztell64 OF ((gzFile))`

7.655.3.4 `ZEXTERN z_off64_t ZEXPORT gzseek64 OF ((gzFile, z_off64_t, int))`

7.655.3.5 `ZEXTERN gzFile ZEXPORT gzopen64 OF ((const char *, const char *))`

7.655.3.6 `void free OF ((voidpf ptr))`

7.655.3.7 `voidp malloc OF ((uInt size))`

## 7.656 src/main/decaf/internal/util/zip/inffast.h File Reference

### Functions

- void ZLIB\_INTERNAL inflate\_fast OF ((z\_stream strm, unsigned start))

### 7.656.1 Function Documentation

- 7.656.1.1 void ZLIB\_INTERNAL inflate\_fast OF ((z\_stream strm, unsigned start))

## 7.657 src/main/decaf/internal/util/zip/inffixed.h File Reference

## 7.658 src/main/decaf/internal/util/zip/inflate.h File Reference

### Data Structures

- struct `inflate_state`

### Defines

- `#define GUNZIP`

### Enumerations

- enum `inflate_mode` {  
    **HEAD**, **FLAGS**, **TIME**, **OS**,  
    **EXLEN**, **EXTRA**, **NAME**, **COMMENT**,  
    **HCRC**, **DICTID**, **DICT**, **TYPE**,  
    **TYPEDO**, **STORED**, **COPY\_**, **COPY**,  
    **TABLE**, **LENLENS**, **CODELENS**, **LEN\_**,  
    **LEN**, **LENEXT**, **DIST**, **DISTEXT**,  
    **MATCH**, **LIT**, **CHECK**, **LENGTH**,  
    **DONE**, **BAD**, **MEM**, **SYNC** }

#### 7.658.1 Define Documentation

##### 7.658.1.1 `#define GUNZIP`

#### 7.658.2 Enumeration Type Documentation

##### 7.658.2.1 enum `inflate_mode`

Enumerator:

***HEAD***  
***FLAGS***  
***TIME***  
***OS***  
***EXLEN***  
***EXTRA***  
***NAME***  
***COMMENT***  
***HCRC***  
***DICTID***  
***DICT***  
***TYPE***

*TYPEDO*  
*STORED*  
*COPY\_*  
*COPY*  
*TABLE*  
*LENLENS*  
*CODELENS*  
*LEN\_*  
*LEN*  
*LENEXT*  
*DIST*  
*DISTEXT*  
*MATCH*  
*LIT*  
*CHECK*  
*LENGTH*  
*DONE*  
*BAD*  
*MEM*  
*SYNC*

## 7.659 src/main/decaf/internal/util/zip/inftrees.h File Reference

### Data Structures

- struct **code**

### Defines

- #define **ENOUGH\_LENS** 852
- #define **ENOUGH\_DISTS** 592
- #define **ENOUGH** (ENOUGH\_LENS+ENOUGH\_DISTS)

### Enumerations

- enum **codetype** { **CODES**, **LENS**, **DISTS** }

### Functions

- int ZLIB\_INTERNAL inflate\_table **OF** ((**codetype** type, unsigned short FAR \*lens, unsigned codes, **code** FAR \*FAR \*table, unsigned FAR \*bits, unsigned short FAR \*work))

#### 7.659.1 Define Documentation

7.659.1.1 #define **ENOUGH** (ENOUGH\_LENS+ENOUGH\_DISTS)

7.659.1.2 #define **ENOUGH\_DISTS** 592

7.659.1.3 #define **ENOUGH\_LENS** 852

#### 7.659.2 Enumeration Type Documentation

7.659.2.1 enum **codetype**

Enumerator:

***CODES***

***LENS***

***DISTS***

#### 7.659.3 Function Documentation

7.659.3.1 int ZLIB\_INTERNAL inflate\_table **OF** ((**codetype** type, unsigned short FAR \*lens, unsigned codes, **code** FAR \*FAR \*table, unsigned FAR \*bits, unsigned short FAR \*work))





```

21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
}

```

### 7.660.1.3 local const int base\_\_dist[D\_CODES]

Initial value:

```

{
    0,      1,      2,      3,      4,      6,      8,      12,     16,     24,
   32,     48,     64,     96,    128,    192,    256,    384,    512,    768,
  1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
}

```

### 7.660.1.4 local const int base\_\_length[LENGTH\_CODES]

Initial value:

```

{
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}

```

### 7.660.1.5 local const ct\_\_data static \_\_dtree[D\_CODES]

Initial value:

```

{
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}

```

### 7.660.1.6 local const ct\_\_data static \_\_ltree[L\_CODES+2]

## 7.661 src/main/decaf/internal/util/zip/zconf.h File Reference

```
#include <decaf/util/Config.h>
```

### Defines

- `#define const`
- `#define MAX__MEM__LEVEL 9`
- `#define MAX__WBITS 15`
- `#define OF(args) ()`
- `#define ZEXTERN extern`
- `#define SEEK__SET 0`
- `#define SEEK__CUR 1`
- `#define SEEK__END 2`
- `#define z__off__t long`
- `#define z__off64__t z__off__t`

### Typedefs

- `typedef unsigned char Byte`
- `typedef unsigned int uInt`
- `typedef unsigned long uLong`
- `typedef Byte FAR Bytef`
- `typedef char FAR charf`
- `typedef int FAR intf`
- `typedef uInt FAR uIntf`
- `typedef uLong FAR uLongf`
- `typedef Byte const * voidpc`
- `typedef Byte FAR * voidpf`
- `typedef Byte * voidp`

## 7.661.1 Define Documentation

- 7.661.1.1 `#define const`
- 7.661.1.2 `#define MAX__MEM__LEVEL 9`
- 7.661.1.3 `#define MAX__WBITS 15`
- 7.661.1.4 `#define OF(args) ()`
- 7.661.1.5 `#define SEEK__CUR 1`
- 7.661.1.6 `#define SEEK__END 2`
- 7.661.1.7 `#define SEEK__SET 0`
- 7.661.1.8 `#define z__off64__t z__off__t`
- 7.661.1.9 `#define z__off__t long`
- 7.661.1.10 `#define ZEXTERN extern`

## 7.661.2 Typedef Documentation

- 7.661.2.1 `typedef unsigned char Byte`
- 7.661.2.2 `typedef Byte FAR Bytef`
- 7.661.2.3 `typedef char FAR charf`
- 7.661.2.4 `typedef int FAR intf`
- 7.661.2.5 `typedef unsigned int uInt`
- 7.661.2.6 `typedef uInt FAR uIntf`
- 7.661.2.7 `typedef unsigned long uLong`
- 7.661.2.8 `typedef uLong FAR uLongf`
- 7.661.2.9 `typedef Byte* voidp`
- 7.661.2.10 `typedef Byte const* voidpc`
- 7.661.2.11 `typedef Byte FAR* voidpf`

## 7.662 src/main/decaf/internal/util/zip/zlib.h File Reference

```
#include "zconf.h"
```

### Data Structures

- struct `z_stream_s`
- struct `gz_header_s`
- struct `internal_state`

### Defines

- `#define ZLIB_VERSION "1.2.5"`
- `#define ZLIB_VERNUM 0x1250`
- `#define ZLIB_VER_MAJOR 1`
- `#define ZLIB_VER_MINOR 2`
- `#define ZLIB_VER_REVISION 5`
- `#define ZLIB_VER_SUBREVISION 0`
- `#define Z_NO_FLUSH 0`
- `#define Z_PARTIAL_FLUSH 1`
- `#define Z_SYNC_FLUSH 2`
- `#define Z_FULL_FLUSH 3`
- `#define Z_FINISH 4`
- `#define Z_BLOCK 5`
- `#define Z_TREES 6`
- `#define Z_OK 0`
- `#define Z_STREAM_END 1`
- `#define Z_NEED_DICT 2`
- `#define Z_ERRNO (-1)`
- `#define Z_STREAM_ERROR (-2)`
- `#define Z_DATA_ERROR (-3)`
- `#define Z_MEM_ERROR (-4)`
- `#define Z_BUF_ERROR (-5)`
- `#define Z_VERSION_ERROR (-6)`
- `#define Z_NO_COMPRESSION 0`
- `#define Z_BEST_SPEED 1`
- `#define Z_BEST_COMPRESSION 9`
- `#define Z_DEFAULT_COMPRESSION (-1)`
- `#define Z_FILTERED 1`
- `#define Z_HUFFMAN_ONLY 2`
- `#define Z_RLE 3`
- `#define Z_FIXED 4`
- `#define Z_DEFAULT_STRATEGY 0`
- `#define Z_BINARY 0`
- `#define Z_TEXT 1`
- `#define Z_ASCII Z_TEXT`
- `#define Z_UNKNOWN 2`
- `#define Z_DEFLATED 8`

- `#define Z_NULL 0`
- `#define zlib_version zlibVersion()`
- `#define deflateInit(strm, level) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))`
- `#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`
- `#define deflateInit2(strm, level, method, windowBits, memLevel, strategy)`
- `#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`
- `#define inflateBackInit(strm, windowBits, window)`

## Typedefs

- `typedef voidpf alloc_func OF ((voidpf opaque, uInt items, uInt size))`
- `typedef struct z_stream_s z_stream`
- `typedef z_stream FAR * z_streamp`
- `typedef struct gz_header_s gz_header`
- `typedef gz_header FAR * gz_headerp`
- `typedef voidp gzFile`

## Functions

- `ZEXTERN const char *ZEXPORT zlibVersion OF ((void))`
- `ZEXTERN int ZEXPORT deflate OF ((z_streamp strm, int flush))`
- `ZEXTERN int ZEXPORT deflateEnd OF ((z_streamp strm))`
- `ZEXTERN int ZEXPORT deflateSetDictionary OF ((z_streamp strm, const Bytef *dictionary, uIntdictLength))`
- `ZEXTERN int ZEXPORT deflateCopy OF ((z_streamp dest, z_streamp source))`
- `ZEXTERN int ZEXPORT deflateParams OF ((z_streamp strm, int level, int strategy))`
- `ZEXTERN int ZEXPORT deflateTune OF ((z_streamp strm, int good_length, int max_lazy, int nice_length, int max_chain))`
- `ZEXTERN uLong ZEXPORT deflateBound OF ((z_streamp strm, uLong sourceLen))`
- `ZEXTERN int ZEXPORT deflatePrime OF ((z_streamp strm, int bits, int value))`
- `ZEXTERN int ZEXPORT deflateSetHeader OF ((z_streamp strm, gz_headerp head))`
- `ZEXTERN int ZEXPORT inflateReset2 OF ((z_streamp strm, int windowBits))`
- `ZEXTERN int ZEXPORT inflateBack OF ((z_streamp strm, in_func in, void FAR *in_desc, out_func out, void FAR *out_desc))`
- `ZEXTERN int ZEXPORT compress OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen))`
- `ZEXTERN int ZEXPORT compress2 OF ((Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen, int level))`
- `ZEXTERN uLong ZEXPORT compressBound OF ((uLong sourceLen))`
- `ZEXTERN gzFile ZEXPORT gzdopen OF ((int fd, const char *mode))`
- `ZEXTERN int ZEXPORT gzbuffer OF ((gzFile file, unsigned size))`
- `ZEXTERN int ZEXPORT gzsetparams OF ((gzFile file, int level, int strategy))`
- `ZEXTERN int ZEXPORT gzread OF ((gzFile file, voidp buf, unsigned len))`
- `ZEXTERN int ZEXPORT gzwrite OF ((gzFile file, voidpc buf, unsigned len))`
- `ZEXTERN int ZEXPORTVA gzprintf OF ((gzFile file, const char *format,...))`
- `ZEXTERN int ZEXPORT gzputs OF ((gzFile file, const char *s))`
- `ZEXTERN char *ZEXPORT gzgets OF ((gzFile file, char *buf, int len))`
- `ZEXTERN int ZEXPORT gzputc OF ((gzFile file, int c))`

- ZEXTERN int ZEXPORT gzgetc **OF** ((**gzFile** file))
- ZEXTERN int ZEXPORT gzungetc **OF** ((int c, **gzFile** file))
- ZEXTERN int ZEXPORT gzflush **OF** ((**gzFile** file, int flush))
- ZEXTERN const char \*ZEXPORT gzerror **OF** ((**gzFile** file, int \*errnum))
- ZEXTERN **uLong** ZEXPORT Adler32 **OF** ((**uLong** Adler, const **Bytef** \*buf, **uInt** len))
- ZEXTERN **uLong** ZEXPORT Crc32 **OF** ((**uLong** Crc, const **Bytef** \*buf, **uInt** len))
- ZEXTERN int ZEXPORT deflateInit\_ **OF** ((**z\_stream** strm, int level, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit\_ **OF** ((**z\_stream** strm, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT deflateInit2\_ **OF** ((**z\_stream** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit2\_ **OF** ((**z\_stream** strm, intwindowBits, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateBackInit\_ **OF** ((**z\_stream** strm, int windowBits, unsigned char FAR \*window, const char \*version, int stream\_size))
- ZEXTERN **gzFile** ZEXPORT gzopen **OF** ((const char \*, const char \*))
- ZEXTERN **z\_off\_t** ZEXPORT gzseek **OF** ((**gzFile**, **z\_off\_t**, int))
- ZEXTERN **z\_off\_t** ZEXPORT gztell **OF** ((**gzFile**))
- ZEXTERN **uLong** ZEXPORT Adler32\_combine **OF** ((**uLong**, **uLong**, **z\_off\_t**))
- ZEXTERN const char \*ZEXPORT zError **OF** ((int))
- ZEXTERN int ZEXPORT inflateSyncPoint **OF** ((**z\_stream**))
- ZEXTERN int ZEXPORT inflateUndermine **OF** ((**z\_stream**, int))

## 7.662.1 Define Documentation

**7.662.1.1** `#define deflateInit(strm, level) deflateInit_((strm), (level),  
ZLIB_VERSION, sizeof(z_stream))`

**7.662.1.2** `#define deflateInit2(strm, level, method, windowBits, memLevel,  
strategy)`

Value:

```
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), \
              (strategy), ZLIB_VERSION, sizeof(z_stream))
```

**7.662.1.3** `#define inflateBackInit(strm, windowBits, window)`

Value:

```
inflateBackInit_((strm), (windowBits), (window), \
                 ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.4 `#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.5 `#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.6 `#define Z_ASCII Z_TEXT`

7.662.1.7 `#define Z_BEST_COMPRESSION 9`

7.662.1.8 `#define Z_BEST_SPEED 1`

7.662.1.9 `#define Z_BINARY 0`

7.662.1.10 `#define Z_BLOCK 5`

7.662.1.11 `#define Z_BUF_ERROR (-5)`

7.662.1.12 `#define Z_DATA_ERROR (-3)`

7.662.1.13 `#define Z_DEFAULT_COMPRESSION (-1)`

7.662.1.14 `#define Z_DEFAULT_STRATEGY 0`

7.662.1.15 `#define Z_DEFLATED 8`

7.662.1.16 `#define Z_ERRNO (-1)`

7.662.1.17 `#define Z_FILTERED 1`

7.662.1.18 `#define Z_FINISH 4`

7.662.1.19 `#define Z_FIXED 4`

7.662.1.20 `#define Z_FULL_FLUSH 3`

7.662.1.21 `#define Z_HUFFMAN_ONLY 2`

7.662.1.22 `#define Z_MEM_ERROR (-4)`

7.662.1.23 `#define Z_NEED_DICT 2`

7.662.1.24 `#define Z_NO_COMPRESSION 0`

7.662.1.25 `#define Z_NO_FLUSH 0`

7.662.1.26 `#define Z_NULL 0`

7.662.1.27 `#define Z_OK 0`

7.662.1.28 `#define Z_PARTIAL_FLUSH 1`

7.662.1.29 `#define Z_RLE 3`

7.662.1.30 `#define Z_STREAM_END 1`

7.662.1.31 `#define Z_STREAM_ERROR (-2)`

7.662.1.32 `#define Z_SYNC_FLUSH 2`

7.662.1.33 `#define Z_TEXT 1`



## 7.663 src/main/decaf/internal/util/zip/zutil.h File Reference

```
#include "zlib.h"
```

### Defines

- `#define ZLIB_INTERNAL`
- `#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]`
- `#define ERR_RETURN(strm, err) return (strm->msg = (char*)ERR_MSG(err), (err))`
- `#define DEF_WBITS MAX_WBITS`
- `#define DEF_MEM_LEVEL 8`
- `#define STORED_BLOCK 0`
- `#define STATIC_TREES 1`
- `#define DYN_TREES 2`
- `#define MIN_MATCH 3`
- `#define MAX_MATCH 258`
- `#define PRESET_DICT 0x20`
- `#define OS_CODE 0x03`
- `#define F_OPEN(name, mode) fopen((name), (mode))`
- `#define Assert(cond, msg)`
- `#define Trace(x)`
- `#define Tracev(x)`
- `#define Tracevv(x)`
- `#define Tracec(c, x)`
- `#define Tracecv(c, x)`
- `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`

### Typedefs

- `typedef unsigned char uch`
- `typedef uch FAR uchf`
- `typedef unsigned short ush`
- `typedef ush FAR ushf`
- `typedef unsigned long ulg`

### Functions

- `ZEXTERN uLong ZEXPORT adler32_combine64 OF ((uLong, uLong, z_off_t)`
- `void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, uInt len))`
- `int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, uInt len))`
- `void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, uInt len))`
- `voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))`
- `void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))`

## Variables

- `const char *const z_errmsg [10]`

## 7.663.1 Define Documentation

- 7.663.1.1 `#define Assert(cond, msg)`
- 7.663.1.2 `#define DEF_MEM_LEVEL 8`
- 7.663.1.3 `#define DEF_WBITS MAX_WBITS`
- 7.663.1.4 `#define DYN_TREES 2`
- 7.663.1.5 `#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]`
- 7.663.1.6 `#define ERR_RETURN(strm, err) return (strm->msg = (char*)ERR_MSG(err), (err))`
- 7.663.1.7 `#define F_OPEN(name, mode) fopen((name), (mode))`
- 7.663.1.8 `#define MAX_MATCH 258`
- 7.663.1.9 `#define MIN_MATCH 3`
- 7.663.1.10 `#define OS_CODE 0x03`
- 7.663.1.11 `#define PRESET_DICT 0x20`
- 7.663.1.12 `#define STATIC_TREES 1`
- 7.663.1.13 `#define STORED_BLOCK 0`
- 7.663.1.14 `#define Trace(x)`
- 7.663.1.15 `#define Tracec(c, x)`
- 7.663.1.16 `#define Tracecv(c, x)`
- 7.663.1.17 `#define Tracev(x)`
- 7.663.1.18 `#define Tracevv(x)`
- 7.663.1.19 `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`
- 7.663.1.20 `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- 7.663.1.21 `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- 7.663.1.22 `#define ZLIB_INTERNAL`

## 7.663.2 Typedef Documentation

- 7.663.2.1 `typedef unsigned char uch`
- 7.663.2.2 `typedef uch FAR uchf`
- 7.663.2.3 `typedef unsigned long ulg`
- 7.663.2.4 `typedef unsigned short ush`
- 7.663.2.5 `typedef ush FAR ushf`

## 7.664 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <vector>
```

### Data Structures

- class **decaf::io::BlockingByteArrayInputStream**  
*This is a blocking version of a byte buffer stream.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.665 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::io::BufferedInputStream**

*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of **io** (p. 154) operations on the input stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.666 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::io::BufferedOutputStream**

*Wrapper around another output stream that buffers output before writing to the target output stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.667 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

### Data Structures

- class **decaf::io::ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 1020) contains an **internal** (p. 144) buffer that contains bytes that may be read from the stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.668 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <utility>
```

### Data Structures

- class **decaf::io::ByteArrayOutputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**



## 7.669 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::DataInput**

*The **DataInput** (p. 1558) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.670 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::DataInputStream**

*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.671 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::DataOutput**

*The **DataOutput** (p. 1574) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.672 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::io::DataOutputStream**

*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.673 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::EOFException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.674 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::io::FileDescriptor**

*This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.675 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::FilterInputStream**

A ***FilterInputStream*** (p. 1894) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.676 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::FilterOutputStream**

*This class is the superclass of all classes that filter output streams.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**



## 7.677 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::Flushable**

*A **Flushable** (p. 1936) is a destination of data that can be flushed.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.678 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::io::InputStream**

*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.679 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/Reader.h>
```

### Data Structures

- class **decaf::io::InputStreamReader**

*An **InputStreamReader** (p. 2053) is a bridge from byte streams to character streams.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.680 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::InterruptedIOException**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.681 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::io::IOException**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.682 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::OutputStream**

*Base interface for any class that wants to represent an output stream of bytes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.683 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Writer.h>
```

### Data Structures

- class **decaf::io::OutputStreamWriter**  
*A class for turning a character stream into a byte stream.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.684 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/FilterInputStream.h>
```

### Data Structures

- class **decaf::io::PushbackInputStream**

*A **PushbackInputStream** (p. 3141) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**



## 7.685 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/lang/Readable.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **decaf::io::Reader**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.686 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::UnsupportedEncodingException**  
*Thrown when the the Character Encoding is not supported.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.687 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::UTFDataFormatException**

*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.688 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <vector>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/lang/Appendable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::Writer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.689 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Appendable**

*An object to which char sequences and values can be appended.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.690 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/System.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

### Data Structures

- class **decaf::lang::ArrayPointer**< T, REFCOUNTER >  
*Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used.*
- struct **decaf::lang::ArrayPointer**< T, REFCOUNTER >::**ArrayData**
- class **decaf::lang::ArrayPointerComparator**< T, R >  
*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 730).*
- struct **std::less**< **decaf::lang::ArrayPointer**< T > >  
*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **std**

### Functions

- template<typename T , typename R , typename U >  
 bool **decaf::lang::operator**== (const ArrayPointer< T, R > &left, const U \*right)
- template<typename T , typename R , typename U >  
 bool **decaf::lang::operator**== (const U \*left, const ArrayPointer< T, R > &right)
- template<typename T , typename R , typename U >  
 bool **decaf::lang::operator**!= (const ArrayPointer< T, R > &left, const U \*right)

- template<typename T , typename R , typename U >  
bool **decaf::lang::operator!=** (const U \*left, const ArrayPointer< T, R > &right)

## 7.691 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Boolean**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**



## 7.692 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Byte**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.693 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Character**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.694 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::CharSequence**

*A **CharSequence** (p. 1135) is a readable sequence of char values.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.695 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Comparable**< **T** >

*This interface imposes a total ordering on the objects of each class that implements it.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.696 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Double**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.697 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

### Data Structures

- class **decaf::lang::Exception**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.698 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::ClassCastException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.699 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**



## 7.700 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.701 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.702 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.703 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::InterruptedException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.704 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.705 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::NullPointerException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.707 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**



## 7.708 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::RuntimeException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.709 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Float**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.710 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

### Data Structures

- class **decaf::lang::Integer**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.711 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
```

### Data Structures

- class **decaf::lang::Iterable**< **E** >

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2152) type for generic collections API calls.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.712 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Long**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.713 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Math**

*The class **Math** (p. 2497) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.714 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2835) is the superclass of classes **Byte** (p. 960), **Double** (p. 1788), **Float** (p. 1904), **Integer** (p. 2077), **Long** (p. 2420), and **Short** (p. 3440).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.715 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

### Data Structures

- struct **decaf::lang::STATIC\_CAST\_TOKEN**
- struct **decaf::lang::DYNAMIC\_CAST\_TOKEN**
- class **decaf::lang::Pointer**< T, REFCOUNTER >
 

*Decaf's implementation of a Smart **Pointer** (p. 2946) that is a template on a Type and is **Thread** (p. 3765) Safe if the default Reference Counter is used.*
- class **decaf::lang::PointerComparator**< T, R >
 

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2946) instance.*
- struct **std::less**< **decaf::lang::Pointer**< T > >
 

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **std**

### Functions

- template<typename T, typename R, typename U >
 bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >
 bool **decaf::lang::operator==** (const U \*left, const Pointer< T, R > &right)
- template<typename T, typename R, typename U >
 bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >
 bool **decaf::lang::operator!=** (const U \*left, const Pointer< T, R > &right)



## 7.716 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::lang::Readable**  
*A **Readable** (p. 3160) is a source of characters.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**
- namespace **decaf::lang**

## 7.717 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Runnable**

*Interface for a runnable object - defines a task that can be run by a thread.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.718 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Runtime**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.719 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Short**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.720 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

### Data Structures

- class **decaf::lang::String**

*The **String** (p. 3665) class represents an immutable sequence of chars.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.721 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

### Data Structures

- class **decaf::lang::System**

*The **System** (p. 3726) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.722 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Thread**  
*A **Thread** (p. 3765) is a concurrent unit of execution.*
- class **decaf::lang::Thread::UncaughtExceptionHandler**  
*Interface for handlers invoked when a **Thread** (p. 3765) abruptly terminates due to an uncaught exception.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

## 7.723 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::ThreadGroup**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**



## 7.724 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Throwable**  
*This class represents an error that has occurred.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.725 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::net::BindException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.726 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::net::ConnectException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.727 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::HttpRetryException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.728 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/net/InetAddress.h>
```

### Data Structures

- class **decaf::net::Inet4Address**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.729 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

### Data Structures

- class **decaf::net::Inet6Address**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.730 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/ArrayPointer.h>
```

### Data Structures

- class **decaf::net::InetAddress**  
*Represents an IP address.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.731 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketAddress.h>
#include <string>
```

### Data Structures

- class **decaf::net::InetSocketAddress**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**



## 7.732 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::MalformedURLException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.733 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::net::NoRouteToHostException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.734 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::net::PortUnreachableException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.735 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::ProtocolException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.736 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
#include <string>
```

### Data Structures

- class **decaf::net::ServerSocket**  
*This class implements server sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.737 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

### Data Structures

- class **decaf::net::ServerSocketFactory**  
*Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.738 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::Socket**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.739 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketAddress**  
*Base class for protocol specific **Socket** (p. 3503) addresses.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**



## 7.740 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
```

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketError**

*Static utility class to simplify handling of error codes for socket operations.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.741 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::SocketException**  
*Exception for errors when manipulating sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.742 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/net/UnknownHostException.h>
```

### Data Structures

- class **decaf::net::SocketFactory**

*The **SocketFactory** (p. 3524) is used to create **Socket** (p. 3503) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.743 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/FileDescriptor.h>
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/net/SocketOptions.h>
#include <string>
```

### Data Structures

- class **decaf::net::SocketImpl**  
*Acts as a base class for all physical **Socket** (p. 3503) implementations.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.744 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketImplFactory**  
*Factory class interface for a Factory that creates ScketImpl objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.745 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketOptions**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.746 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InterruptedIOException.h>
```

### Data Structures

- class **decaf::net::SocketTimeoutException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.747 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

### Data Structures

- class **decaf::net::ssl::SSLContext**

*Represents on implementation of the Secure **Socket** (p. 3503) Layer for streaming based sockets.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**



## 7.748 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandom.h>
```

### Data Structures

- class **decaf::net::ssl::SSLContextSpi**  
*Defines the interface that should be provided by an **SSLContext** (p. 3545) provider.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.749 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::net::ssl::SSLParameters**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.750 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocket.h>
```

### Data Structures

- class **decaf::net::ssl::SSLServerSocket**  
*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.751 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
#include <vector>
#include <string>
```

### Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**  
*Factory class interface that provides methods to create SSL Server Sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.752 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Socket.h>
#include <decaf/net/ssl/SSLParameters.h>
```

### Data Structures

- class **decaf::net::ssl::SSLSocket**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.753 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
#include <vector>
#include <string>
```

### Data Structures

- class **decaf::net::ssl::SSLSocketFactory**  
*Factory class interface for a **SocketFactory** (p. 3524) that can create **SSLSocket** (p. 3563) objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.754 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::UnknownHostException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.755 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::UnknownServiceException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**



## 7.756 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

### Data Structures

- class **decaf::net::URI**

*This class represents an instance of a **URI** (p. 3921) as defined by RFC 2396.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.757 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::net::URISyntaxException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.758 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

### Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 3960) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.759 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

### Data Structures

- class **decaf::net::URLDecoder**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.760 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

### Data Structures

- class **decaf::net::URLEncoder**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.761 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/InvalidMarkException.h>
```

### Data Structures

- class **decaf::nio::Buffer**  
*A container for data of a specific primitive type.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.762 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::nio::BufferOverflowException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.763 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::nio::BufferUnderflowException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**



## 7.764 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::ByteBuffer**

*This class defines six categories of operations upon byte buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.765 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

### Data Structures

- class **decaf::nio::CharBuffer**

*This class defines four categories of operations upon character buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.766 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::DoubleBuffer**

*This class defines four categories of operations upon double buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.767 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::FloatBuffer**

*This class defines four categories of operations upon float buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.768 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::IntBuffer**

*This class defines four categories of operations upon int buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.769 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **decaf::nio::InvalidMarkException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.770 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::LongBuffer**

*This class defines four categories of operations upon long long buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.771 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

### Data Structures

- class **decaf::nio::ReadOnlyBufferException**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**



## 7.772 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::ShortBuffer**

*This class defines four categories of operations upon short buffers:.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.773 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

### Data Structures

- class `decaf::security::auth::x500::X500Principal`

### Namespaces

- namespace `decaf`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `decaf::security`
- namespace `decaf::security::auth`
- namespace `decaf::security::auth::x500`

## 7.774 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::Certificate**  
*Base interface for all identity certificates.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.775 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateEncodingException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.776 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.777 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateExpiredException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.778 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-*  
*ments.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.779 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateParsingException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**



## 7.780 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

### Data Structures

- class **decaf::security::cert::X509Certificate**  
*Base interface for all identity certificates.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.781 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::security::GeneralSecurityException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.782 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

### Data Structures

- class **decaf::security::InvalidKeyException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.783 src/main/decaf/security/Key.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::security::Key**  
*The **Key** (p. 2293) interface is the top-level interface for all keys.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.784 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::KeyException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.785 src/main/decaf/security/KeyManagementException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/KeyException.h>
```

### Data Structures

- class **decaf::security::KeyManagementException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.786 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::NoSuchAlgorithmException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.787 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::NoSuchProviderException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**



## 7.788 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::security::Principal**

*Base interface for a principal, which can represent an individual or organization.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::security**

## 7.789 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
```

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::security::PublicKey**  
*A public key.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.790 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Random.h>
#include <decaf/security/SecureRandomSpi.h>
#include <memory>
```

### Data Structures

- class **decaf::security::SecureRandom**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.791 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::security::SecureRandomSpi**

*Interface class used by Security Service Providers to implement a source of secure random bytes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::security**

## 7.792 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::SignatureException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.793 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 1184) interface, to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.794 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractList**< **E** >

*This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.795 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractMap**< K, V, COMPARATOR >

*This class provides a skeletal implementation of the **Map** (p. 2459) interface, to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**



## 7.796 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractQueue**< **E** >  
*This class provides skeletal implementations of some **Queue** (p. 3149) operations.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.797 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractSequentialList**< E >

*This class provides a skeletal implementation of the **List** (p. 2337) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.798 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractSet**< E >

*This class provides a skeletal implementation of the **Set** (p. 3439) interface to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.799 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

### Data Structures

- class **decaf::util::Collection< E >**  
*The root interface in the collection hierarchy.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.800 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

### Data Structures

- class **decaf::util::Comparator**< **T** >  
*A comparison function, which imposes a total ordering on some collection of objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.801 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

### Data Structures

- class **decaf::util::comparators::Less**< **E** >  
*Simple **Less** (p. 2328) **Comparator** (p. 1217) that compares to elements to determine if the first is less than the second.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::comparators**

## 7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**

*A boolean value that may be updated atomically.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.803 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**  
*An int value that may be updated atomically.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**



## 7.804 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.805 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Long.h>
#include <apr_atomic.h>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**  
*An Pointer reference that may be updated atomically.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.806 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

### Data Structures

- class **decaf::util::concurrent::BlockingQueue**< E >

A **decaf::util::Queue** (p. 3149) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.807 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.808 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::Callable< V >**  
*A task that returns a result and may throw an exception.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.809 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::CancellationException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.810 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

### Defines

- `#define WAIT_INFINITE 0xFFFFFFFF`  
*The synchronized macro defines a mechanism for synchronizing a section of **code** (p. 1183).*
- `#define synchronized(W)`

#### 7.810.1 Define Documentation

##### 7.810.1.1 `#define synchronized(W)`

**Value:**

```
if(false){}
else
    for( decaf::util::concurrent::Lock lock_W(W);
        lock_W.isLocked(); lock_W.unlock() )
```

##### 7.810.1.2 `#define WAIT_INFINITE 0xFFFFFFFF`

The synchronized macro defines a mechanism for synchronizing a section of **code** (p. 1183). The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 4893) { // Do something that needs synchronizing. } }
```

## 7.811 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< **K**, **V**, **COMPARATOR** >  
*Interface for a **Map** (p. 2459) type that provides additional **atomic** (p. 173) **putIfAbsent**, **remove**, and **replace** methods alongside the already available **Map** (p. 2459) interface.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**



## 7.812 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR >  
*Map* (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.813 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::CountDownLatch**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.814 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

### Data Structures

- class **decaf::util::concurrent::Delayed**

*A mix-in style interface for marking objects that should be acted upon after a given delay.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.815 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::ExecutionException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.816 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

### Data Structures

- class **decaf::util::concurrent::Executor**  
*An object that executes submitted **decaf.lang.Runnable** (p. 3325) tasks.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.817 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

### Data Structures

- class **decaf::util::concurrent::ExecutorService**

*An **Executor** (p. 1869) that provides methods to manage termination and methods that can produce a **Future** (p. 1966) for tracking progress of one or more asynchronous tasks.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.818 src/main/decaf/util/concurrent/Future.h File Reference

### Data Structures

- class **decaf::util::concurrent::Future**< V >  
*A **Future** (p. 1966) represents the result of an asynchronous computation.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.819 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::Lock**

*A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**



## 7.820 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

### Data Structures

- class **decaf::util::concurrent::locks::Lock**

***Lock** (p. 2377) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.821 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

### Data Structures

- class **decaf::util::concurrent::locks::Condition**

***Condition** (p. 1249) factors out the **Mutex** (p. 2782) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2377) implementations.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.822 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

*Basic thread blocking primitives for creating **locks** (p. 174) and other synchronization classes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.823 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

### Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

*A **ReadWriteLock** (p. 3172) maintains a pair of associated **locks** (p. 174), one for read-only operations and one for writing.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.824 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**  
*A reentrant mutual exclusion **Lock** (p. 2377) with extended capabilities.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.825 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 2782) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.826 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class `decaf::util::concurrent::PooledThread`

### Namespaces

- namespace `decaf`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `decaf::util`
- namespace `decaf::util::concurrent`

## 7.827 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::PooledThreadListener**  
*Abstract Listener Interface for users of `ThreadPool` (p. 3777).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**



## 7.828 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.829 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

### Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**  
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p.??).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.830 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```

### Data Structures

- class **decaf::util::concurrent::Semaphore**  
*A counting semaphore.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.831 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::Synchronizable**  
*The interface for all synchronizable objects (that is, objects that can be locked and unlocked).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.832 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

### Data Structures

- class **decaf::util::concurrent::SynchronousQueue< E >**  
*A **blocking queue** (p. 843) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*
- class **decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.833 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::TaskListener**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.834 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ThreadFactory**  
*public interface **ThreadFactory** (p. 3774)*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.835 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

### Data Structures

- class **decaf::util::concurrent::ThreadPool**

*Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**



## 7.836 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::TimeoutException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.837 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **decaf::util::concurrent::TimeUnit**

*A **TimeUnit** (p. 3807) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.838 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

### Data Structures

- class **decaf::util::Date**  
*Wrapper class around a time value in milliseconds.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.839 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

### Data Structures

- class **decaf::util::Iterator**< T >  
*Defines an object that can be used to iterate over the elements of a collection.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.840 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

### Data Structures

- class **decaf::util::List**< **E** >  
*An ordered collection (also known as a sequence).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.841 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::ListIterator**< **E** >

*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.842 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/util/logging/SimpleFormatter.h>
#include <decaf/io/IOException.h>
#include <decaf/internal/io/StandardErrorOutputStream.h>
```

### Data Structures

- class **decaf::util::logging::ConsoleHandler**  
*This **Handler** (p. 1978) publishes log records to `System.err`.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.843 src/main/decaf/util/logging/EventManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::EventManager**

***ErrorManager** (p. 1828) objects can be attached to *Handlers* to process any error that occur on a *Handler* (p. 1978) during Logging.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**



## 7.844 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::Filter**

*A **Filter** (p. 1893) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.845 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Handler.h>
```

### Data Structures

- class **decaf::util::logging::Formatter**  
*A **Formatter** (p. 1964) provides support for formatting LogRecords.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.846 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/Level.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::Handler**

A **Handler** (p. 1978) object takes log messages from a **Logger** (p. 2386) and exports them.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.847 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::logging::Level**

*The **Level** (p. 2332) class defines a set of standard **logging** (p. 175) levels that can be used to control **logging** (p. 175) output.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.848 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/LogManager.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <list>
#include <string>
#include <stdarg.h>
```

### Data Structures

- class **decaf::util::logging::Logger**

A *Logger* (p. 2386) object is used to log messages for a specific system or application component.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.849 src/main/decaf/util/logging/LoggerCommon.h File Reference

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

### Enumerations

- enum **decaf::util::logging::Levels** {  
    **decaf::util::logging::Off**, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**,  
    **decaf::util::logging::Debug**,  
    **decaf::util::logging::Info**, **decaf::util::logging::Warn**, **decaf::util::logging::Error**,  
    **decaf::util::logging::Fatal**,  
    **decaf::util::logging::Throwing** }  
*Defines an enumeration for logging levels.*

## 7.850 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

### Defines

- **#define LOGDECAF\_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;**
- **#define LOGDECAF\_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);**
- **#define LOGDECAF\_DECLARE\_LOCAL(loggerName) decaf::util::logging::Logger loggerName;**
- **#define LOGDECAF\_DEBUG(logger, message) logger.debug(\_\_FILE\_\_, \_\_LINE\_\_, message);**
- **#define LOGDECAF\_DEBUG\_1(logger, message, value)**
- **#define LOGDECAF\_INFO(logger, message) logger.info(\_\_FILE\_\_, \_\_LINE\_\_, message);**
- **#define LOGDECAF\_ERROR(logger, message) logger.error(\_\_FILE\_\_, \_\_LINE\_\_, message);**
- **#define LOGDECAF\_WARN(logger, message) logger.warn(\_\_FILE\_\_, \_\_LINE\_\_, message);**
- **#define LOGDECAF\_FATAL(logger, message) logger.fatal(\_\_FILE\_\_, \_\_LINE\_\_, message);**

### 7.850.1 Define Documentation

**7.850.1.1 #define LOGDECAF\_DEBUG(logger, message) logger.debug(\_\_FILE\_\_, \_\_LINE\_\_, message);**

**7.850.1.2 #define LOGDECAF\_DEBUG\_1(logger, message, value)**

**Value:**

```
;
{
    std::ostringstream ostream;
    ostream << message << value;
    logger.debug(__FILE__, __LINE__, ostream.str());
}
```

- 7.850.1.3 `#define LOGDECAF_DECLARE(loggerName) static  
decaf::util::logging::SimpleLogger loggerName;`
- 7.850.1.4 `#define LOGDECAF_DECLARE_LOCAL(loggerName) de-  
caf::util::logging::Logger loggerName;`
- 7.850.1.5 `#define LOGDECAF_ERROR(logger, message) logger.error(__-  
FILE __, __LINE __, message);`
- 7.850.1.6 `#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE __-  
__, __LINE __, message);`
- 7.850.1.7 `#define LOGDECAF_INFO(logger, message) logger.info(__FILE __,  
__LINE __, message);`
- 7.850.1.8 `#define LOGDECAF_INITIALIZE(loggerName,  
className, loggerFamily) decaf::util::logging::SimpleLogger  
className::loggerName(loggerFamily);`
- 7.850.1.9 `#define LOGDECAF_WARN(logger, message) logger.warn(__FILE __-  
__, __LINE __, message);`



## 7.851 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

### Data Structures

- class `decaf::util::logging::LoggerHierarchy`

### Namespaces

- namespace `decaf`

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace `decaf::util`
- namespace `decaf::util::logging`

## 7.852 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 2406) object that is used to maintain a set of shared state about Loggers and log services.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.853 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Level.h>
#include <decaf/util/Config.h>
#include <memory>
#include <string>
```

### Data Structures

- class **decaf::util::logging::LogRecord**  
*LogRecord* (p. 2413) objects are used to pass *logging* (p. 175) requests between the *logging* (p. 175) framework and individual log Handlers.

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.854 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::util::logging::LogWriter**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.855 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

### Data Structures

- class **decaf::util::logging::MarkBlockLogger**

*Defines a class that can be used to mark the entry and exit from scoped blocks.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.856 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 3126).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.857 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2413) in a human readable format.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.858 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::SimpleLogger**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**



## 7.859 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::StreamHandler**  
*Stream based **logging** (p. 175) **Handler** (p. 1978).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.860 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::XMLFormatter**  
*Format a **LogRecord** (p. 2413) into a standard XML format.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.861 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::util::Map**< **K**, **V**, **COMPARATOR** >  
***Map** (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map**< **K**, **V**, **COMPARATOR** >::**Entry**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.862 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

### Data Structures

- class **decaf::util::PriorityQueue< E >**  
*An unbounded priority queue based on a binary heap algorithm.*
- class **decaf::util::PriorityQueue< E >::PriorityQueueIterator**
- class **decaf::util::PriorityQueue< E >::ConstPriorityQueueIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.863 src/main/decaf/util/Properties.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::util::Properties**  
*Java-like properties class for mapping string names to string values.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**
- namespace **decaf::util**

## 7.864 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

### Data Structures

- class **decaf::util::Random**

***Random** (p. 3155) Value Generator which is used to generate a stream of pseudorandom numbers.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.865 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

### Data Structures

- class **decaf::util::Set**< **E** >  
*A collection that contains no duplicate elements.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.866 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

### Data Structures

- class **decaf::util::StlList< E >**  
*List* (p. 2337) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.
- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**



## 7.867 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

### Data Structures

- class **decaf::util::StlMap< K, V, COMPARATOR >**  
*Map* (p. 2459) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.868 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::StlQueue< T >**  
*The **Queue** (p. 3149) class accepts messages with an **psuh(m)** command where *m* is the message to be queued.*
- class **decaf::util::StlQueue< T >::QueueIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.869 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

### Data Structures

- class **decaf::util::StlSet< E >**  
*Set (p. 3439) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.*
- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.870 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::util::StringTokenizer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.871 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::Timer**

*A facility for threads to schedule tasks for future execution in a background thread.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.872 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3790).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

## 7.873 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

### Data Structures

- class **decaf::util::UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3969)).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.874 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

### Data Structures

- class **decaf::util::zip::Adler32**

*Clas that can be used to compute an Adler-32 **Checksum** (p. 1142) for a data stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**



## 7.875 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterInputStream.h>
```

### Data Structures

- class **decaf::util::zip::CheckedInputStream**

*An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1142) of the bytes read, the **Checksum** (p. 1142) can then be used to verify the integrity of the input stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.876 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterOutputStream.h>
```

### Data Structures

- class **decaf::util::zip::CheckedOutputStream**

*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1142) of the bytes written, the **Checksum** (p. 1142) can then be used to verify the integrity of the output stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.877 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::Checksum**

*An interface used to represent **Checksum** (p. 1142) values in the Zip package.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.878 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

### Data Structures

- class **decaf::util::zip::CRC32**

*Class that can be used to compute a CRC-32 checksum for a data stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.879 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::zip::DataFormatException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.880 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::Deflater**

*This class compresses data using the DEFLATE algorithm (see [specification](#)).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.881 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Deflater.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

*Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.882 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/zip/DataFormatException.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::Inflater**

*This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**



## 7.883 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Inflater.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::InflaterInputStream**  
*A FilterInputStream that decompresses data read from the wrapped InputStream instance.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.884 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::util::zip::ZipException**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

# Index

- ~AbstractCollection
  - decaf::util::AbstractCollection, 182
- ~AbstractList
  - decaf::util::AbstractList, 192
- ~AbstractMap
  - decaf::util::AbstractMap, 193
- ~AbstractQueue
  - decaf::util::AbstractQueue, 195
- ~AbstractSequentialList
  - decaf::util::AbstractSequentialList, 198
- ~AbstractSet
  - decaf::util::AbstractSet, 199
- ~AbstractTransportFactory
  - activemq::transport::AbstractTransportFactory, 201
- ~ActiveMQAckHandler
  - activemq::core::ActiveMQAckHandler, 203
- ~ActiveMQBlobMessage
  - activemq::commands::ActiveMQBlobMessage, 205
- ~ActiveMQBlobMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 214
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 222
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 210
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 218
  - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 226
  - activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 230
- ~ActiveMQBytesMessage
  - activemq::commands::ActiveMQBytesMessage, 236
- ~ActiveMQBytesMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 254
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 270
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 250
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 258
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 262
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 266
- ~ActiveMQCPP
  - activemq::library::ActiveMQCPP, 320
- ~ActiveMQConnection
  - activemq::core::ActiveMQConnection, 278
- ~ActiveMQConnectionFactory
  - activemq::core::ActiveMQConnectionFactory, 294
- ~ActiveMQConnectionMetaData
  - activemq::core::ActiveMQConnectionMetaData, 304
- ~ActiveMQConsumer
  - activemq::core::ActiveMQConsumer, 312
- ~ActiveMQDestination
  - activemq::commands::ActiveMQDestination, 324
- ~ActiveMQDestinationMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 338
  - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 340
  - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 341
  - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 342
  - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 343
  - activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 344
- ~ActiveMQException
  - activemq::exceptions::ActiveMQException, 358
- ~ActiveMQMapMessage
  - activemq::commands::ActiveMQMapMessage, 363
- ~ActiveMQMapMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 379
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 391
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 375

- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 383
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 387
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 395
- ~ActiveMQMessage
  - activemq::commands::ActiveMQMessage, 398
- ~ActiveMQMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 406
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 418
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 402
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 410
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 414
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 422
- ~ActiveMQMessageTemplate
  - activemq::commands::ActiveMQMessageTemplate, 428
- ~ActiveMQObjectMessage
  - activemq::commands::ActiveMQObjectMessage, 444
- ~ActiveMQObjectMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 451
  - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 463
  - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 447
  - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 455
  - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 459
  - activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 467
- ~ActiveMQProducer
  - activemq::core::ActiveMQProducer, 472
- ~ActiveMQProperties
  - activemq::util::ActiveMQProperties, 479
- ~ActiveMQQueue
  - activemq::commands::ActiveMQQueue, 484
- ~ActiveMQQueueBrowser
  - activemq::core::ActiveMQQueueBrowser, 488
- ~ActiveMQQueueMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 496
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 508
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 492
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 500
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 504
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 512
- ~ActiveMQSession
  - activemq::core::ActiveMQSession, 519
- ~ActiveMQSessionExecutor
  - activemq::core::ActiveMQSessionExecutor, 534
- ~ActiveMQStreamMessage
  - activemq::commands::ActiveMQStreamMessage, 540
- ~ActiveMQStreamMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 557
  - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 569
  - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 553
  - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 561
  - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 565
  - activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 573
- ~ActiveMQTempDestination
  - activemq::commands::ActiveMQTempDestination, 577
- ~ActiveMQTempDestinationMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 585
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 597
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 581
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 589
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 593
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 601
- ~ActiveMQTempQueue
  - activemq::commands::ActiveMQTempQueue, 605
- ~ActiveMQTempQueueMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 614

- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 626
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 610
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 618
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 622
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 630
- ~ActiveMQTempTopic
  - activemq::commands::ActiveMQTempTopic, 634
- ~ActiveMQTempTopicMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 647
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 655
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 639
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 643
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 651
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 659
- ~ActiveMQTextMessage
  - activemq::commands::ActiveMQTextMessage, 663
- ~ActiveMQTextMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 676
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 688
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 668
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 672
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 680
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 684
- ~ActiveMQTopic
  - activemq::commands::ActiveMQTopic, 692
- ~ActiveMQTopicMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 704
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 716
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 696
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 700
- ActiveMQTempQueueMarshaller, 708
- ActiveMQTempQueueMarshaller, 712
- ActiveMQTempQueueMarshaller, 720
- ActiveMQTempQueueMarshaller, 723
- ~Appendable
  - decaf::lang::Appendable, 726
- ~AprPool
  - decaf::internal::AprPool, 728
- ~ArrayPointer
  - decaf::lang::ArrayPointer, 733
- ~AtomicBoolean
  - decaf::util::concurrent::atomic::AtomicBoolean, 738
- ~AtomicInteger
  - decaf::util::concurrent::atomic::AtomicInteger, 742
- ~AtomicReference
  - decaf::util::concurrent::atomic::AtomicReference, 747
- ~BackupTransport
  - activemq::transport::failover::BackupTransport, 752
- ~BackupTransportPool
  - activemq::transport::failover::BackupTransportPool, 756
- ~BaseCommand
  - activemq::commands::BaseCommand, 759
- ~BaseCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 780
  - activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 801
  - activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 780
  - activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 801
  - activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 801
  - activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 801
- ~BaseDataStructure
  - activemq::wireformat::openwire::marshal::BaseDataStructure, 831

- ~BindException
  - decaf::net::BindException, 837
- ~BlockingByteArrayInputStream
  - decaf::io::BlockingByteArrayInputStream, 840
- ~BlockingQueue
  - decaf::util::concurrent::BlockingQueue, 846
- ~Boolean
  - decaf::lang::Boolean, 851
- ~BooleanExpression
  - activemq::commands::BooleanExpression, 855
- ~BooleanStream
  - activemq::wireformat::openwire::utils::BooleanStream, 858
- ~BrokenBarrierException
  - decaf::util::concurrent::BrokenBarrierException, 861
- ~BrokerError
  - activemq::commands::BrokerError, 864
- ~BrokerException
  - activemq::exceptions::BrokerException, 867
- ~BrokerId
  - activemq::commands::BrokerId, 870
- ~BrokerIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 881
  - activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 893
  - activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 873
  - activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 877
  - activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 885
  - activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 889
- ~BrokerInfo
  - activemq::commands::BrokerInfo, 897
- ~BrokerInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 913
  - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 925
  - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 905
  - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 909
  - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 917
  - activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 921
- ~Buffer
  - decaf::nio::Buffer, 930
- ~BufferFactory
  - decaf::internal::nio::BufferFactory, 944
- ~BufferOverflowException
  - decaf::nio::BufferOverflowException, 955
- ~BufferUnderflowException
  - decaf::nio::BufferUnderflowException, 958
- ~BufferedInputStream
  - decaf::io::BufferedInputStream, 936
- ~BufferedOutputStream
  - decaf::io::BufferedOutputStream, 941
- ~Byte
  - decaf::lang::Byte, 962
- ~ByteArrayAdapter
  - decaf::internal::util::ByteArrayAdapter, 976
- ~ByteArrayBuffer
  - decaf::internal::nio::ByteArrayBuffer, 1003
- ~ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 1023
- ~ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 1029
- ~ByteBuffer
  - decaf::nio::ByteBuffer, 1036
- ~BytesMessage
  - cms::BytesMessage, 1059
- ~CMSEncryptionSupport
  - cms::CMSEncryptionSupport, 1161
- ~CMSEncryptionSupport
  - activemq::util::CMSEncryptionSupport, 1163
- ~CMSProperties
  - cms::CMSProperties, 1166
- ~CMSSecurityException
  - cms::CMSSecurityException, 1169
- ~CRC32
  - decaf::util::CRC32, 1525
- ~CachedConsumer
  - activemq::cmsutil::CachedConsumer, 1072
- ~CachedProducer
  - activemq::cmsutil::CachedProducer, 1076
- ~Callable
  - decaf::util::concurrent::Callable, 1082
- ~CancellationException
  - decaf::util::concurrent::CancellationException, 1084
- ~Certificate
  - decaf::security::cert::Certificate, 1087
- ~CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1091
- ~CertificateException
  - decaf::security::cert::CertificateException, 1093

- ~CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1095
- ~CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1097
- ~CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1099
- ~CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1113
- ~CharBuffer
  - decaf::nio::CharBuffer, 1122
- ~CharSequence
  - decaf::lang::CharSequence, 1135
- ~CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1138
- ~CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1141
- ~Checksum
  - decaf::util::zip::Checksum, 1142
- ~ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1146
- ~CloseTransportsTask
  - activemq::transport::failover::CloseTransportsTask, 1151
- ~Closeable
  - cms::Closeable, 1148
  - decaf::io::Closeable, 1149
- ~CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1154
- ~CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1158
- ~CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1173
- ~Collection
  - decaf::util::Collection, 1186
- ~Command
  - activemq::commands::Command, 1195
- ~CommandVisitor
  - activemq::state::CommandVisitor, 1202
- ~CommandVisitorAdapter
  - activemq::state::CommandVisitorAdapter, 1211
- ~Comparable
  - decaf::lang::Comparable, 1214
- ~Comparator
  - decaf::util::Comparator, 1217
- ~CompositeData
  - activemq::util::CompositeData, 1220
- ~CompositeTask
  - activemq::threads::CompositeTask, 1221
  - decaf::security::cert::CertificateExpiredException, 1095
  - activemq::threads::CompositeTaskRunner, 1224
  - activemq::threads::CompositeTaskRunner, 1224
  - activemq::transport::CompositeTransport, 1226
  - activemq::transport::CompositeTransport, 1226
  - decaf::util::concurrent::ConcurrentMap, 1229
  - decaf::util::concurrent::ConcurrentMap, 1229
  - ~ConcurrentStlMap
    - decaf::util::concurrent::ConcurrentStlMap, 1237
  - ~Condition
    - decaf::util::concurrent::locks::Condition, 1251
  - ~ConditionHandle
    - decaf::util::concurrent::ConditionHandle, 1255
  - ~ConnectException
    - decaf::net::ConnectException, 1260
  - ~Connection
    - cms::Connection, 1263
  - ~ConnectionControl
    - activemq::commands::ConnectionControl, 1268
  - ~ConnectionControlMarshaller
    - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1281
    - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1293
    - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1273
    - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1277
    - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1285
    - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1289
  - ~ConnectionError
    - activemq::commands::ConnectionError, 1297
  - ~ConnectionErrorMarshaller
    - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1313
    - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1301
    - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1305
    - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1309
    - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1317

- activemq::wireformat::openwire::marshal::v6::ConnectionFactoryMarshaller, 1415
- activemq::wireformat::openwire::marshal::v4::ConsumerControl, 1415
- 1321
- ~ConnectionFactory
  - cms::ConnectionFactory, 1325
- ~ConnectionId
  - activemq::commands::ConnectionId, 1328
- ~ConnectionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1431
  - 1344
  - ~ConsumerIdMarshaller
  - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1448
  - 1332
  - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1436
  - 1336
  - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1440
  - 1340
  - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1444
  - 1348
  - activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1452
  - 1352
- ~ConnectionInfo
  - activemq::commands::ConnectionInfo, 1456
  - 1356
  - ~ConsumerInfo
- ~ConnectionInfoMarshaller
  - activemq::commands::ConsumerInfo, 1461
  - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1451
  - 1374
  - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1459
  - 1362
  - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1469
  - 1366
  - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1473
  - 1370
  - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1477
  - 1378
  - activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1485
  - 1382
- ~ConnectionMetaData
  - cms::ConnectionMetaData, 1386
- ~ConsumerState
  - activemq::state::ConsumerState, 1492
- ~ControlCommand
  - activemq::commands::ControlCommand, 1494
- ~ControlCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1510
- ~ConsumerControl
  - activemq::commands::ConsumerControl, 1498
  - 1402
  - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1502
- ~ConsumerControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1506
  - 1419
  - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1514
  - 1407
  - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1518
  - 1411



- ~CountDownLatch
  - decaf::util::concurrent::CountDownLatch, 1521
- ~DataArrayResponse
  - activemq::commands::DataArrayResponse, 1529
- ~DataArrayResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1544
  - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1532
  - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1536
  - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1540
  - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1548
  - activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1552
- ~DataFormatException
  - decaf::util::zip::DataFormatException, 1556
- ~DataInput
  - decaf::io::DataInput, 1559
- ~DataInputStream
  - decaf::io::DataInputStream, 1567
- ~DataOutput
  - decaf::io::DataOutput, 1575
- ~DataOutputStream
  - decaf::io::DataOutputStream, 1580
- ~DataResponse
  - activemq::commands::DataResponse, 1584
- ~DataResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1607
  - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1595
  - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1599
  - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1603
  - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1587
  - activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1591
- ~DataStreamMarshaller
  - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1611
- ~DataStructure
  - activemq::commands::DataStructure, 1660
- ~Date
  - decaf::util::Date, 1666
- ~DecafRuntime
  - decaf::internal::DecafRuntime, 1670
- ~DedicatedTaskRunner
  - activemq::threads::DedicatedTaskRunner, 1671
- ~DefaultPrefetchPolicy
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
- ~DefaultRedeliveryPolicy
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
- ~DefaultSSLContextMarshaller
  - decaf::internal::net::ssl::DefaultSSLContext, 1684
- ~DefaultSSLServerSocketFactory
  - activemq::core::policies::DefaultSSLServerSocketFactory, 1694
- ~DefaultSSLSocketFactory
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1698
- ~DefaultServerSocketFactory
  - decaf::internal::net::DefaultServerSocketFactory, 1683
- ~DefaultSocketFactory
  - decaf::internal::net::DefaultSocketFactory, 1688
- ~DefaultTransportListener
  - activemq::transport::DefaultTransportListener, 1704
- ~Deflater
  - decaf::util::zip::Deflater, 1708
- ~DeflaterOutputStream
  - decaf::util::zip::DeflaterOutputStream, 1718
- ~Delayed
  - activemq::core::Delayed, 1720
- ~DeliveryMode
  - activemq::core::DeliveryMode, 1722
- ~Destination
  - activemq::core::Destination, 1724
- ~DestinationInfo
  - activemq::core::DestinationInfo, 1728
- ~DestinationInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1715
  - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1733
  - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1737
  - activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1741
  - activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1753
  - activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1749

- ~DestinationResolver
  - activemq::cmsutil::DestinationResolver, 1756
- ~DiscoveryEvent
  - activemq::commands::DiscoveryEvent, 1759
- ~DiscoveryEventMarshaller
  - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1779
  - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1767
  - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1771
  - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1775
  - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1783
  - activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1763
- ~Dispatcher
  - activemq::core::Dispatcher, 1787
- ~Double
  - decaf::lang::Double, 1790
- ~DoubleArrayBuffer
  - decaf::internal::nio::DoubleArrayBuffer, 1803
- ~DoubleBuffer
  - decaf::nio::DoubleBuffer, 1811
- ~DynamicDestinationResolver
  - activemq::cmsutil::DynamicDestinationResolver, 1822
- ~EOFException
  - decaf::io::EOFException, 1826
- ~Entry
  - decaf::util::Map::Entry, 1824
- ~ErrorManager
  - decaf::util::logging::ErrorManager, 1829
- ~Exception
  - decaf::lang::Exception, 1833
- ~ExceptionListener
  - cms::ExceptionListener, 1838
- ~ExceptionResponse
  - activemq::commands::ExceptionResponse, 1840
- ~ExceptionResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1863
  - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1847
  - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1851
  - activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1859
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1855
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1843
- ~ExecutionException
  - decaf::util::concurrent::ExecutionException, 1867
- ~Executor
  - decaf::util::concurrent::Executor, 1870
- ~ExecutorService
  - decaf::util::concurrent::ExecutorService, 1872
- ~FailoverTransport
  - activemq::transport::failover::FailoverTransport, 1875
- ~FailoverTransportFactory
  - activemq::transport::failover::FailoverTransportFactory, 1886
- ~FailoverTransportListener
  - activemq::transport::failover::FailoverTransportListener, 1889
- ~FileDescriptor
  - decaf::io::FileDescriptor, 1892
- ~Filter
  - decaf::util::logging::Filter, 1893
- ~FilterInputStream
  - decaf::io::FilterInputStream, 1896
- ~FilterOutputStream
  - decaf::io::FilterOutputStream, 1901
- ~Float
  - decaf::lang::Float, 1906
- ~FloatArrayBuffer
  - decaf::internal::nio::FloatArrayBuffer, 1919
- ~FloatBuffer
  - decaf::nio::FloatBuffer, 1927
- ~FlushCommand
  - activemq::commands::FlushCommand, 1938
- ~FlushCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1957
  - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1945
  - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1949
  - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1953
  - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1965
  - activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1945
- ~Flushable
  - decaf::util::Flushable, 1936
- ~Formatter

- decaf::util::logging::Formatter, 1964
- ~Future
  - decaf::util::concurrent::Future, 1967
- ~FutureResponse
  - activemq::transport::correlator::FutureResponse, 1969
- ~GeneralSecurityException
  - decaf::security::GeneralSecurityException, 1972
- ~GenericResource
  - decaf::internal::util::GenericResource, 1974
- ~Handler
  - decaf::util::logging::Handler, 1979
- ~HexStringParser
  - decaf::internal::util::HexStringParser, 1982
- ~HexTable
  - activemq::wireformat::openwire::utils::HexTable, 1984
- ~HttpRequestException
  - decaf::net::HttpRequestException, 1987
- ~IOException
  - decaf::io::IOException, 2143
- ~IOTransport
  - activemq::transport::IOTransport, 2147
- ~IdGenerator
  - activemq::util::IdGenerator, 1989
- ~IllegalArgumentException
  - decaf::lang::exceptions::IllegalArgumentException, 1992
- ~IllegalMonitorStateException
  - decaf::lang::exceptions::IllegalMonitorStateException, 1995
- ~IllegalStateException
  - cms::IllegalStateException, 1997
  - decaf::lang::exceptions::IllegalStateException, 1999
- ~IllegalThreadStateException
  - decaf::lang::exceptions::IllegalThreadStateException, 2002
- ~InactivityMonitor
  - activemq::transport::inactivity::InactivityMonitor, 2005
- ~IndexOutOfBoundsException
  - decaf::lang::exceptions::IndexOutOfBoundsException, 2009
- ~Inet4Address
  - decaf::net::Inet4Address, 2012
- ~Inet6Address
  - decaf::net::Inet6Address, 2015
- ~InetAddress
  - decaf::net::InetAddress, 2018
- ~InetSocketAddress
  - decaf::net::InetSocketAddress, 2023
- ~Inflater
  - decaf::util::zip::Inflater, 2029
- ~InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 2038
- ~InputStream
  - decaf::io::InputStream, 2045
- ~InputStreamReader
  - decaf::io::InputStreamReader, 2054
- ~IntArrayBuffer
  - decaf::internal::nio::IntArrayBuffer, 2060
- ~IntBuffer
  - decaf::nio::IntBuffer, 2068
- ~Integer
  - decaf::lang::Integer, 2080
- ~IntegerResponse
  - activemq::commands::IntegerResponse, 2092
- ~IntegerResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2111
  - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2099
  - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2103
  - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2107
  - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2115
  - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2095
- ~InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2122
- ~InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2125
- ~InterruptedIOException
  - decaf::io::InterruptedIOException, 2128
- ~InvalidClientIdException
  - cms::InvalidClientIdException, 2130
- ~InvalidDestinationException
  - cms::InvalidDestinationException, 2131
- ~InvalidKeyException
  - decaf::security::InvalidKeyException, 2133
- ~InvalidMarkException
  - decaf::nio::InvalidMarkException, 2136
- ~InvalidSelectorException
  - cms::InvalidSelectorException, 2138
- ~InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2140
- ~Iterable
  - decaf::lang::Iterable, 2152
- ~Iterator
  - decaf::util::Iterator, 2154

- ~JournalQueueAck
  - activemq::commands::JournalQueueAck, 2157
- ~JournalQueueAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2180
  - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2164
  - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2172
  - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2176
  - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2168
  - activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2160
- ~JournalTopicAck
  - activemq::commands::JournalTopicAck, 2184
- ~JournalTopicAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2209
  - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2193
  - activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2197
  - activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2205
  - activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2189
  - activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2201
- ~JournalTrace
  - activemq::commands::JournalTrace, 2213
- ~JournalTraceMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2232
  - activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2216
  - activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2220
  - activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2228
  - activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2236
  - activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2224
- ~JournalTransaction
  - activemq::commands::JournalTransaction, 2240
- ~JournalTransactionMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2263
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2247
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2251
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2259
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2255
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2243
- ~KeepAliveInfo
  - activemq::commands::KeepAliveInfo, 2267
- ~KeepAliveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2290
  - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2274
  - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2278
  - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2282
  - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2286
  - activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2270
- ~Key
  - decaf::security::Key, 2294
- ~KeyException
  - decaf::security::KeyException, 2296
- ~KeyManagementException
  - decaf::security::KeyManagementException, 2299
- ~LastPartialCommand
  - activemq::commands::LastPartialCommand, 2301
- ~LastPartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2325
  - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2313
  - activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2309
  - activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2321
  - activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2317
  - activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2305
- ~Less
  - decaf::util::comparators::Less, 2328
- ~Level
  - decaf::util::logging::Level, 2334
- ~List
  - decaf::util::List, 2338

- ~ListIterator
  - decaf::util::ListIterator, 2345
- ~LocalTransactionId
  - activemq::commands::LocalTransactionId, 2348
- ~LocalTransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2372
  - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2356
  - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2360
  - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2368
  - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2364
  - activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2352
- ~Lock
  - decaf::util::concurrent::Lock, 2375
  - decaf::util::concurrent::locks::Lock, 2378
- ~LockSupport
  - decaf::util::concurrent::locks::LockSupport, 2384
- ~LogManager
  - decaf::util::logging::LogManager, 2408
- ~LogRecord
  - decaf::util::logging::LogRecord, 2414
- ~LogWriter
  - decaf::util::logging::LogWriter, 2418
- ~Logger
  - decaf::util::logging::Logger, 2389
- ~LoggerHierarchy
  - decaf::util::logging::LoggerHierarchy, 2398
- ~LoggingInputStream
  - activemq::io::LoggingInputStream, 2399
- ~LoggingOutputStream
  - activemq::io::LoggingOutputStream, 2401
- ~LoggingTransport
  - activemq::transport::logging::LoggingTransport, 2403
- ~Long
  - decaf::lang::Long, 2423
- ~LongArrayBuffer
  - decaf::internal::nio::LongArrayBuffer, 2438
- ~LongBuffer
  - decaf::nio::LongBuffer, 2446
- ~LongSequenceGenerator
  - activemq::util::LongSequenceGenerator, 2455
- ~MalformedURLException
  - decaf::net::MalformedURLException, 2457
- ~Map
  - decaf::util::Map, 2460
- ~MapMessage
  - cms::MapMessage, 2474
- ~MarkBlockLogger
  - decaf::util::logging::MarkBlockLogger, 2483
- ~MarshalAware
  - activemq::wireformat::openwire::marshal::v1::MarshalAware, 2484
- ~MarshallerFactory
  - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2491
  - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2492
  - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2488
  - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2489
  - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2490
  - activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2487
- ~MarshallSupport
  - activemq::util::MarshallSupport, 2494
- ~Math
  - decaf::lang::Math, 2499
- ~MemoryUsage
  - activemq::util::MemoryUsage, 2513
- ~Message
  - activemq::commands::Message, 2520
  - cms::Message, 2538
- ~MessageAck
  - activemq::commands::MessageAck, 2560
- ~MessageAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2582
  - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2570
  - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2574
  - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2578
  - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2586
  - activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2566
- ~MessageConsumer
  - cms::MessageConsumer, 2590
- ~MessageCreator
  - activemq::cmsutil::MessageCreator, 2593
- ~MessageDispatch
  - activemq::commands::MessageDispatch, 2595
- ~MessageDispatchChannel
  - activemq::core::MessageDispatchChannel, 2600

- ~MessageDispatchMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2623
  - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2607
  - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2611
  - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2619
  - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2615
  - activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2627
- ~MessageDispatchNotification
  - activemq::commands::MessageDispatchNotification, 2631
- ~MessageDispatchNotificationMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2652
  - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2640
  - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2644
  - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2648
  - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2656
  - activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2636
- ~MessageEOFException
  - cms::MessageEOFException, 2661
- ~MessageEnumeration
  - cms::MessageEnumeration, 2659
- ~MessageFormatException
  - cms::MessageFormatException, 2662
- ~MessageId
  - activemq::commands::MessageId, 2664
- ~MessageIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2689
  - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2669
  - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2681
  - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2673
  - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2677
  - activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2685
- ~MessageListener
  - cms::MessageListener, 2692
- ~MessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2714
  - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2704
  - activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2699
  - activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2709
  - activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2694
  - activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2719
- ~MessageNotReadableException
  - cms::MessageNotReadableException, 2723
- ~MessageNotWritableException
  - cms::MessageNotWritableException, 2724
- ~MessageProducer
  - cms::MessageProducer, 2726
- ~MessagePropertyInterceptor
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2733
- ~MessagePull
  - activemq::commands::MessagePull, 2740
- ~MessagePullMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2761
  - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2745
  - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2753
  - activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2757
  - activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2749
  - activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2765
- ~MockTransport
  - activemq::transport::mock::MockTransport, 2770
- ~MockTransportFactory
  - activemq::transport::mock::MockTransportFactory, 2780
- ~Mutex
  - cms::util::concurrent::Mutex, 2783
- ~MutexHandle
  - cms::util::concurrent::MutexHandle, 2787
- ~Network
  - activemq::net::Network, 2791
- ~NetworkBridgeFilter
  - activemq::commands::NetworkBridgeFilter, 2794
- ~NetworkBridgeFilterMarshaller
  - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2817

- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2797
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2809
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2813
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2805
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2801
- ~NoRouteToHostException
  - decaf::net::NoRouteToHostException, 2821
- ~NoSuchAlgorithmException
  - decaf::security::NoSuchAlgorithmException, 2824
- ~NoSuchElementException
  - decaf::lang::exceptions::NoSuchElementException, 2827
- ~NoSuchProviderException
  - decaf::security::NoSuchProviderException, 2830
- ~NullPointerException
  - decaf::lang::exceptions::NullPointerException, 2833
- ~Number
  - decaf::lang::Number, 2835
- ~NumberFormatException
  - decaf::lang::exceptions::NumberFormatException, 2840
- ~ObjectMessage
  - cms::ObjectMessage, 2841
- ~OpenSSLContextSpi
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2843
- ~OpenSSLParameters
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2845
- ~OpenSSLServerSocket
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2849
- ~OpenSSLServerSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2855
- ~OpenSSLSocket
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2862
- ~OpenSSLSocketException
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2873
- ~OpenSSLSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2877
- ~OpenSSLSocketInputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
- ~OpenSSLServerSocket
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2886
- ~OpenWireFormat
  - activemq::wireformat::openwire::OpenWireFormat, 2890
- ~OpenWireFormatFactory
  - activemq::wireformat::openwire::OpenWireFormatFactory, 2899
- ~OpenWireFormatNegotiator
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2902
- ~OpenWireResponseBuilder
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2905
- ~OutputStream
  - decaf::io::OutputStream, 2909
- ~OutputStreamWriter
  - decaf::io::OutputStreamWriter, 2916
- ~PartialCommand
  - activemq::commands::PartialCommand, 2919
- ~PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2943
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2927
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2935
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2939
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2931
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- ~Pointer
  - decaf::lang::Pointer, 2949
- ~PooledSession
  - activemq::cmsutil::PooledSession, 2957
- ~PooledThread
  - decaf::util::concurrent::PooledThread, 2968
- ~PooledThreadListener
  - decaf::util::concurrent::PooledThreadListener, 2971
- ~PortUnreachableException
  - decaf::net::PortUnreachableException, 2974
- ~PrefetchPolicy
  - activemq::core::PrefetchPolicy, 2977
- ~PrimitiveList
  - activemq::util::PrimitiveList, 2982
- ~PrimitiveMap
  - activemq::util::PrimitiveMap, 2984

- activemq::util::PrimitiveMap, 2993
- ~PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3003
- ~PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 3010
- ~PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 3018
- ~Principal
  - decaf::security::Principal, 3026
- ~PriorityQueue
  - decaf::util::PriorityQueue, 3031
- ~ProducerAck
  - activemq::commands::ProducerAck, 3037
- ~ProducerAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3061
  - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 3041
  - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 3049
  - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 3045
  - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3053
  - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3057
- ~ProducerCallback
  - activemq::cmsutil::ProducerCallback, 3064
- ~ProducerExecutor
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3065
- ~ProducerId
  - activemq::commands::ProducerId, 3068
- ~ProducerIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3093
  - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3073
  - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3081
  - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3077
  - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3085
  - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3089
- ~ProducerInfo
  - activemq::commands::ProducerInfo, 3097
- ~ProducerInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3110
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3106
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3118
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3102
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3114
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3122
- ~ProducerState
  - activemq::state::ProducerState, 3125
- ~Properties
  - decaf::util::Properties, 3128
- ~PropertiesChangeListener
  - decaf::util::logging::PropertiesChangeListener, 3135
- ~ProtocolException
  - decaf::net::ProtocolException, 3138
- ~PublicKey
  - decaf::security::PublicKey, 3140
- ~PushbackInputStream
  - decaf::io::PushbackInputStream, 3143
- ~Queue
  - cms::Queue, 3148
  - decaf::util::Queue, 3150
- ~QueueBrowser
  - cms::QueueBrowser, 3153
- ~ReadChecker
  - activemq::transport::inactivity::ReadChecker, 3162
- ~ReadOnlyBufferException
  - decaf::nio::ReadOnlyBufferException, 3170
- ~ReadWriteLock
  - decaf::util::concurrent::locks::ReadWriteLock, 3173
- ~Readable
  - decaf::lang::Readable, 3160
- ~Reader
  - decaf::io::Reader, 3164
- ~ReceiveExecutor
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
- ~RedeliveryPolicy
  - activemq::core::RedeliveryPolicy, 3178
- ~ReentrantLock
  - decaf::util::concurrent::locks::ReentrantLock, 3185
- ~RejectedExecutionException
  - decaf::util::concurrent::RejectedExecutionException, 3190
- ~RejectedExecutionHandler
  - decaf::util::concurrent::RejectedExecutionHandler, 3192



- ~RemoveInfo
  - activemq::commands::RemoveInfo, 3195
- ~RemoveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3211
  - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3199
  - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3207
  - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3219
  - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3215
  - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3203
- ~RemoveSubscriptionInfo
  - activemq::commands::RemoveSubscriptionInfo, 3223
- ~RemoveSubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3228
  - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3236
  - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3232
  - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3248
  - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3244
  - activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3240
- ~ReplayCommand
  - activemq::commands::ReplayCommand, 3252
- ~ReplayCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3259
  - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3263
  - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3267
  - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3255
  - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3275
  - activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3271
- ~ResolveProducerExecutor
  - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3278
- ~ResolveReceiveExecutor
  - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3279
- ~Resource
  - decaf::internal::util::Resource, 3280
- ~ResourceLifecycleManager
  - activemq::cmsutil::ResourceLifecycleManager, 3287
  - decaf::internal::util::ResourceLifecycleManager, 3287
- ~Response
  - activemq::commands::Response, 3286
- ~ResponseBuilder
  - activemq::transport::mock::ResponseBuilder, 3289
- ~ResponseCorrelator
  - activemq::transport::correlator::ResponseCorrelator, 3292
- ~ResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3316
  - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3301
  - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3311
  - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3296
  - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3306
  - activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3321
- ~Runnable
  - decaf::lang::Runnable, 3325
- ~Runtime
  - decaf::lang::Runtime, 3326
- ~RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3329
- ~SSLContext
  - ReplayCommandMarshaller::SSLContext, 3545
- ~SSLContextSpi
  - ReplayCommandMarshaller::SSLContextSpi, 3548
- ~SSLParameters
  - ReplayCommandMarshaller::SSLParameters, 3552
- ~SSLServerSocket
  - ReplayCommandMarshaller::SSLServerSocket, 3556
- ~SSLServerSocketFactory
  - ReplayCommandMarshaller::SSLServerSocketFactory, 3561
- ~SSLServerSocketFactorySpi
  - decaf::net::ssl::SSLServerSocketFactorySpi, 3566
- ~SSLContextSpi
  - decaf::net::ssl::SSLContextSpi, 3572
- ~SecureRandom
  - decaf::security::SecureRandom, 3333
- ~SecureRandomImpl
  - decaf::internal::security::SecureRandomImpl, 3337

- ~SecureRandomSpi
  - decaf::security::SecureRandomSpi, 3339
- ~Semaphore
  - decaf::util::concurrent::Semaphore, 3344
- ~SendExecutor
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3350
- ~ServerSocket
  - decaf::net::ServerSocket, 3355
- ~ServerSocketFactory
  - decaf::net::ServerSocketFactory, 3362
- ~Session
  - cms::Session, 3368
- ~SessionCallback
  - activemq::cmsutil::SessionCallback, 3378
- ~SessionId
  - activemq::commands::SessionId, 3380
- ~SessionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3404
  - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3384
  - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3400
  - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3388
  - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3396
  - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3392
- ~SessionInfo
  - activemq::commands::SessionInfo, 3408
- ~SessionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3420
  - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3428
  - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3424
  - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3432
  - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3416
  - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3412
- ~SessionPool
  - activemq::cmsutil::SessionPool, 3435
- ~SessionState
  - activemq::state::SessionState, 3438
- ~Set
  - decaf::util::Set, 3439
- ~Short
  - decaf::lang::Short, 3442
- ~ShortArrayBuffer
  - decaf::internal::nio::ShortArrayBuffer, 3453
- ~ShortBuffer
  - decaf::nio::ShortBuffer, 3461
- ~ShutdownInfo
  - activemq::commands::ShutdownInfo, 3471
- ~ShutdownInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3482
  - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3478
  - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3490
  - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3494
  - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3486
  - activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3474
- ~SignatureException
  - decaf::security::SignatureException, 3498
- ~SimpleFormatter
  - decaf::util::logging::SimpleFormatter, 3500
- ~SimpleLogger
  - decaf::util::logging::SimpleLogger, 3501
- ~Socket
  - decaf::net::Socket, 3508
- ~SocketAddress
  - decaf::net::SocketAddress, 3519
- ~SocketException
  - decaf::net::SocketException, 3522
- ~SocketFactory
  - decaf::net::SocketFactory, 3525
- ~SocketFileDescriptor
  - decaf::internal::net::SocketFileDescriptor, 3528
- ~SocketImplMarshaller
  - decaf::net::SocketImplMarshaller, 3531
- ~SocketImplFactory
  - decaf::net::SocketImplFactory, 3537
- ~SocketOptions
  - decaf::net::SocketOptions, 3539
- ~SocketTimeoutException
  - decaf::net::SocketTimeoutException, 3543
- ~SslTransport
  - activemq::transport::tcp::SslTransport, 3575
- ~SslTransportFactory
  - activemq::transport::tcp::SslTransportFactory, 3576
- ~StandardErrorOutputStream
  - decaf::internal::io::StandardErrorOutputStream, 3580
- ~StandardInputStream

- decaf::internal::io::StandardInputStream, 3582
- ~StandardOutputStream
  - decaf::internal::io::StandardOutputStream, 3584
- ~Startable
  - cms::Startable, 3586
- ~StaticInitializer
  - activemq::core::ActiveMQConstants::StaticInitializer, 3588
- ~StlList
  - decaf::util::StlList, 3594
- ~StlMap
  - decaf::util::StlMap, 3606
- ~StlQueue
  - decaf::util::StlQueue, 3617
- ~StlSet
  - decaf::util::StlSet, 3625
- ~StompFrame
  - activemq::wireformat::stomp::StompFrame, 3634
- ~StompHelper
  - activemq::wireformat::stomp::StompHelper, 3639
- ~StompWireFormat
  - activemq::wireformat::stomp::StompWireFormat, 3644
- ~StompWireFormatFactory
  - activemq::wireformat::stomp::StompWireFormatFactory, 3647
- ~Stoppable
  - cms::Stoppable, 3648
- ~StreamHandler
  - decaf::util::logging::StreamHandler, 3650
- ~StreamMessage
  - cms::StreamMessage, 3655
- ~String
  - decaf::lang::String, 3666
- ~StringTokenizer
  - decaf::util::StringTokenizer, 3669
- ~SubscriptionInfo
  - activemq::commands::SubscriptionInfo, 3672
- ~SubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3681
  - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3697
  - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3677
  - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3689
  - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3685
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfo, 3693
- ~Synchronizable
  - decaf::util::concurrent::Synchronizable, 3701
- ~SynchronizableImpl
  - decaf::internal::util::concurrent::SynchronizableImpl, 3712
- ~Synchronization
  - activemq::core::Synchronization, 3715
- ~SynchronousQueue
  - decaf::util::concurrent::SynchronousQueue, 3718
- ~System
  - decaf::lang::System, 3728
- ~Task
  - activemq::threads::Task, 3734
- ~TaskListener
  - decaf::util::concurrent::TaskListener, 3735
- ~TaskRunner
  - activemq::threads::TaskRunner, 3736
- ~TcpSocket
  - decaf::internal::net::tcp::TcpSocket, 3741
- ~TcpSocketInputStream
  - decaf::internal::net::tcp::TcpSocketInputStream, 3748
- ~TcpSocketOutputStream
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3750
- ~TcpTransport
  - activemq::transport::tcp::TcpTransport, 3753
- ~TcpTransportFactory
  - activemq::transport::tcp::TcpTransportFactory, 3757
- ~TemporaryQueue
  - cms::TemporaryQueue, 3759
- ~TemporaryTopic
  - cms::TemporaryTopic, 3761
- ~TextMessage
  - cms::TextMessage, 3763
- ~Thread
  - decaf::lang::Thread, 3769
- ~ThreadFactory
  - decaf::util::concurrent::ThreadFactory, 3774
- ~ThreadPool
  - decaf::lang::ThreadGroup, 3776
- ~ThreadPool
  - decaf::util::concurrent::ThreadPool, 3779
- ~Throwable
  - decaf::lang::Throwable, 3784
- ~TimeUnit
  - decaf::util::concurrent::TimeUnit, 3809

- ~TimeoutException
  - decaf::util::concurrent::TimeoutException, 3788
- ~Timer
  - decaf::util::Timer, 3792
- ~TimerTask
  - decaf::util::TimerTask, 3802
- ~TimerTaskHeap
  - decaf::internal::util::TimerTaskHeap, 3805
- ~Topic
  - cms::Topic, 3817
- ~Tracked
  - activemq::state::Tracked, 3818
- ~TransactionId
  - activemq::commands::TransactionId, 3820
- ~TransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3828
  - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3832
  - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3836
  - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3840
  - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3824
  - activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3844
- ~TransactionInfo
  - activemq::commands::TransactionInfo, 3848
- ~TransactionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3856
  - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3872
  - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3860
  - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3868
  - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3852
  - activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3864
- ~TransactionState
  - activemq::state::TransactionState, 3876
- ~TransferQueue
  - decaf::internal::util::concurrent::TransferQueue, 3878
- ~TransferStack
  - decaf::internal::util::concurrent::TransferStack, 3881
- ~Transport
  - activemq::transport::Transport, 3884
- ~TransportFactory
  - activemq::transport::TransportFactory, 3889
- ~TransportFilter
  - activemq::transport::TransportFilter, 3893
- ~TransportListener
  - activemq::transport::TransportListener, 3900
- ~TransportRegistry
  - activemq::transport::TransportRegistry, 3903
- ~URI
  - decaf::net::URI, 3924
- ~URIEncoderDecoder
  - decaf::internal::net::URIEncoderDecoder, 3932
- ~URIHelper
  - decaf::internal::net::URIHelper, 3936
- ~URIPool
  - activemq::transport::failover::URIPool, 3942
- ~URISyntaxException
  - decaf::net::URISyntaxException, 3950
- ~URIType
  - decaf::net::URIType, 3954
- ~URL
  - decaf::net::URL, 3961
- ~URLDecoder
  - decaf::net::URLDecoder, 3962
- ~URLEncoder
  - decaf::net::URLEncoder, 3963
- ~UTFDataFormatException
  - decaf::io::UTFDataFormatException, 3968
- ~UUID
  - decaf::util::UUID, 3971
- ~UncaughtExceptionHandler
  - decaf::util::UncaughtExceptionHandler, 3906
- ~UnknownHostException
  - decaf::net::UnknownHostException, 3908
- ~UnknownServiceException
  - decaf::net::UnknownServiceException, 3911
- ~UnsupportedEncodingException
  - decaf::io::UnsupportedEncodingException, 3915
- ~UnsupportedOperationException
  - cms::UnsupportedOperationException, 3919
  - decaf::lang::exceptions::UnsupportedOperationException, 3917
- ~Usage
  - activemq::util::Usage, 3964
- ~WireFormat

- activemq::wireformat::WireFormat, 3977
- ~WireFormatFactory
  - activemq::wireformat::WireFormatFactory, 3980
- ~WireFormatInfo
  - activemq::commands::WireFormatInfo, 3984
- ~WireFormatInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 4009
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 4001
  - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 4013
  - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4005
  - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3993
  - activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3997
- ~WireFormatNegotiator
  - activemq::wireformat::WireFormatNegotiator, 4016
- ~WireFormatRegistry
  - activemq::wireformat::WireFormatRegistry, 4018
- ~WriteChecker
  - activemq::transport::inactivity::WriteChecker, 4020
- ~Writer
  - decaf::io::Writer, 4022
- ~X500Principal
  - decaf::security::auth::x500::X500Principal, 4027
- ~X509Certificate
  - decaf::security::cert::X509Certificate, 4029
- ~XATransactionId
  - activemq::commands::XATransactionId, 4032
- ~XATransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 4049
  - activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 4041
  - activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 4053
  - activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 4045
  - activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 4057
  - activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4037
- ~XMLFormatter
  - decaf::util::logging::XMLFormatter, 4060
- ~ZipException
  - decaf::util::zip::ZipException, 4065
- \_FALSE
  - decaf::lang::Boolean, 854
- \_TRUE
  - decaf::lang::Boolean, 854
- \_array
  - decaf::internal::nio::CharArrayBuffer, 1118
- \_ByteBuffer
  - decaf::nio::Buffer, 933
- \_deflate
  - deflate.h, 4727
- \_length\_code
  - trees.h, 4735
- \_limit
  - decaf::nio::Buffer, 933
- \_mark
  - decaf::nio::Buffer, 933
- \_markSet
  - decaf::nio::Buffer, 933
- \_position
  - decaf::nio::Buffer, 933
- \_tr\_tally\_dist
  - deflate.h, 4726
- \_tr\_tally\_lit
  - deflate.h, 4726
- ABORT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- abs
  - decaf::lang::Math, 2499, 2500
- AbstractCollection
  - decaf::util::AbstractCollection, 182
- AbstractQueue
  - decaf::util::AbstractQueue, 195
- accept
  - decaf::internal::net::ssl::OpenSSLServerSocket, 2849
  - decaf::internal::net::tcp::TcpSocket, 3741
  - decaf::net::ServerSocket, 3356
  - decaf::net::SocketImpl, 3531
- accepted
  - decaf::net::ServerSocket, 3356
- ACK
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- ACK\_AUTO
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- ACK\_CLIENT

- activemq::wireformat::stomp::StompCommandConstants, 3631
- ACK\_INDIVIDUAL
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- ACK\_TYPE\_CONSUMED
  - activemq::core::ActiveMQConstants, 308
- ACK\_TYPE\_DELIVERED
  - activemq::core::ActiveMQConstants, 308
- ACK\_TYPE\_INDIVIDUAL
  - activemq::core::ActiveMQConstants, 308
- ACK\_TYPE\_POISON
  - activemq::core::ActiveMQConstants, 308
- ACK\_TYPE\_REDELIVERED
  - activemq::core::ActiveMQConstants, 308
- acknowledge
  - activemq::commands::ActiveMQMessageTemplate, 428
  - activemq::core::ActiveMQConsumer, 312
  - activemq::core::ActiveMQSession, 519
  - cms::Message, 2538
- acknowledgeMessage
  - activemq::core::ActiveMQAckHandler, 203
- AcknowledgeMode
  - cms::Session, 3368
- AckType
  - activemq::core::ActiveMQConstants, 308
- ackType
  - activemq::commands::MessageAck, 2564
- acquire
  - decaf::util::concurrent::Semaphore, 3344
- acquireUninterruptibly
  - decaf::util::concurrent::Semaphore, 3345
- action
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3066
- activemq, 85
- activemq/exceptions/ExceptionDefines.h
  - AMQ\_CATCH\_EXCEPTION\_CONVERT, 4170
  - AMQ\_CATCH\_NOTHROW, 4170
  - AMQ\_CATCH\_RETHROW, 4171
  - AMQ\_CATCHALL\_NOTHROW, 4171
  - AMQ\_CATCHALL\_THROW, 4171
- activemq/util/Config.h
  - AMQCPP\_API, 4227
- activemq::cmsutil, 86
- activemq::cmsutil::CachedConsumer, 1071
  - ~CachedConsumer, 1072
  - CachedConsumer, 1072
  - close, 1072
  - getMessageListener, 1072
  - getMessageSelector, 1072
  - operator=, 1072
- activemq::cmsutil::CachedProducer, 1075
  - ~CachedProducer, 1076
  - CachedProducer, 1076
  - close, 1076
  - getDeliveryMode, 1076
  - getDisableMessageID, 1077
  - getDisableMessageTimeStamp, 1077
  - getPriority, 1077
  - getTimeToLive, 1077
  - operator=, 1078
  - send, 1078, 1079
  - setDeliveryMode, 1080
  - setDisableMessageID, 1080
  - setDisableMessageTimeStamp, 1080
  - setPriority, 1080
  - setTimeToLive, 1081
- activemq::cmsutil::CmsAccessor, 1153
  - ~CmsAccessor, 1154
  - checkConnectionFactory, 1154
  - CmsAccessor, 1154
  - createConnection, 1154
  - createSession, 1155
  - destroy, 1155
  - getConnectionFactory, 1155
  - getResourceLifecycleManager, 1155
  - getSessionAcknowledgeMode, 1155
  - init, 1156
  - operator=, 1156
  - setConnectionFactory, 1156
  - setSessionAcknowledgeMode, 1156
- activemq::cmsutil::CmsDestinationAccessor, 1157
  - ~CmsDestinationAccessor, 1158
  - checkDestinationResolver, 1158
  - CmsDestinationAccessor, 1158
  - destroy, 1158
  - getDestinationResolver, 1158
  - init, 1158
  - isPubSubDomain, 1158
  - operator=, 1159
  - resolveDestinationName, 1159
  - setDestinationResolver, 1159
  - setPubSubDomain, 1159
- activemq::cmsutil::CmsTemplate, 1170
  - ~CmsTemplate, 1173
  - CmsTemplate, 1173
  - DEFAULT\_PRIORITY, 1182
  - DEFAULT\_TIME\_TO\_LIVE, 1182
  - destroy, 1173
  - execute, 1174
  - getDefaultDestination, 1175

- getDefaultDestinationName, 1175
- getDeliveryMode, 1175
- getPriority, 1175
- getReceiveTimeout, 1175
- getTimeToLive, 1176
- init, 1176
- isExplicitQosEnabled, 1176
- isMessageIdEnabled, 1176
- isMessageTimestampEnabled, 1176
- isNoLocal, 1176
- operator=, 1176
- ProducerExecutor, 1182
- receive, 1176, 1177
- RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT, 1182
- RECEIVE\_TIMEOUT\_NO\_WAIT, 1182
- ReceiveExecutor, 1182
- receiveSelected, 1177, 1178
- ResolveProducerExecutor, 1182
- ResolveReceiveExecutor, 1182
- send, 1178, 1179
- SendExecutor, 1182
- setDefaultDestination, 1179
- setDefaultDestinationName, 1179
- setDeliveryMode, 1180
- setDeliveryPersistent, 1180
- setExplicitQosEnabled, 1180
- setMessageIdEnabled, 1180
- setMessageTimestampEnabled, 1181
- setNoLocal, 1181
- setPriority, 1181
- setPubSubDomain, 1181
- setReceiveTimeout, 1181
- setTimeToLive, 1181
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 3065
  - ~ProducerExecutor, 3065
  - action, 3066
  - destination, 3066
  - doInCms, 3065
  - getDestination, 3066
  - operator=, 3066
  - parent, 3066
  - ProducerExecutor, 3065
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3174
  - ~ReceiveExecutor, 3175
  - destination, 3176
  - doInCms, 3175
  - getDestination, 3175
  - getMessage, 3175
  - message, 3176
  - noLocal, 3176
  - operator=, 3175
  - parent, 3176
  - ReceiveExecutor, 3175
  - selector, 3176
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3278
  - ~ResolveProducerExecutor, 3278
  - getDestination, 3278
  - operator=, 3278
  - ResolveProducerExecutor, 3278
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3279
  - ~ResolveReceiveExecutor, 3279
  - getDestination, 3279
  - operator=, 3279
  - ResolveReceiveExecutor, 3279
- activemq::cmsutil::CmsTemplate::SendExecutor, 3350
  - ~SendExecutor, 3350
  - doInCms, 3350
  - operator=, 3350
  - SendExecutor, 3350
- activemq::cmsutil::DestinationResolver, 1756
  - ~DestinationResolver, 1756
  - destroy, 1756
  - init, 1756
  - resolveDestinationName, 1757
- activemq::cmsutil::DynamicDestinationResolver, 1821
  - ~DynamicDestinationResolver, 1822
  - destroy, 1822
  - DynamicDestinationResolver, 1822
  - init, 1822
  - operator=, 1822
  - resolveDestinationName, 1822
- activemq::cmsutil::MessageCreator, 2593
  - ~MessageCreator, 2593
  - createMessage, 2593
- activemq::cmsutil::PooledSession, 2955
  - ~PooledSession, 2957
  - close, 2957
  - commit, 2958
  - createBrowser, 2958
  - createBytesMessage, 2959
  - createCachedConsumer, 2959
  - createCachedProducer, 2960
  - createConsumer, 2960, 2961
  - createDurableConsumer, 2961
  - createMapMessage, 2962
  - createMessage, 2962
  - createProducer, 2962
  - createQueue, 2963
  - createStreamMessage, 2963
  - createTemporaryQueue, 2963
  - createTemporaryTopic, 2963

- createTextMessage, 2964
- createTopic, 2964
- getAcknowledgeMode, 2965
- getSession, 2965
- isTransacted, 2965
- operator=, 2965
- PooledSession, 2957
- recover, 2966
- rollback, 2966
- unsubscribe, 2966
- activemq::cmsutil::ProducerCallback, 3064
  - ~ProducerCallback, 3064
  - doInCms, 3064
- activemq::cmsutil::ResourceLifecycleManager, 3282
  - ~ResourceLifecycleManager, 3283
  - addConnection, 3283
  - addDestination, 3283
  - addMessageConsumer, 3283
  - addMessageProducer, 3283
  - addSession, 3283
  - destroy, 3284
  - operator=, 3284
  - releaseAll, 3284
  - ResourceLifecycleManager, 3283
- activemq::cmsutil::SessionCallback, 3378
  - ~SessionCallback, 3378
  - doInCms, 3378
- activemq::cmsutil::SessionPool, 3435
  - ~SessionPool, 3435
  - getResourceLifecycleManager, 3436
  - operator=, 3436
  - returnSession, 3436
  - SessionPool, 3435
  - takeSession, 3436
- activemq::commands, 87
- activemq::commands::ActiveMQBlobMessage, 204
  - ~ActiveMQBlobMessage, 205
  - ActiveMQBlobMessage, 205
  - BINARY\_MIME\_TYPE, 208
  - clone, 205
  - cloneDataStructure, 205
  - copyDataStructure, 205
  - equals, 206
  - getDataStructureType, 206
  - getMimeType, 206
  - getName, 206
  - getRemoteBlobUrl, 206
  - ID\_ACTIVEMQBLOBMESSAGE, 208
  - isDeletedByBroker, 207
  - setDeletedByBroker, 207
  - setMimeType, 207
  - setName, 207
  - setRemoteBlobUrl, 207
  - toString, 207
- activemq::commands::ActiveMQBytesMessage, 233
  - ~ActiveMQBytesMessage, 236
  - ActiveMQBytesMessage, 236
  - clearBody, 236
  - clone, 236
  - cloneDataStructure, 236
  - copyDataStructure, 237
  - equals, 237
  - getBodyBytes, 237
  - getBodyLength, 237
  - getDataStructureType, 238
  - ID\_ACTIVEMQBYTESMESSAGE, 248
  - onSend, 238
  - readBoolean, 238
  - readByte, 238
  - readBytes, 239
  - readChar, 240
  - readDouble, 240
  - readFloat, 241
  - readInt, 241
  - readLong, 241
  - readShort, 242
  - readString, 242
  - readUnsignedShort, 242
  - readUTF, 243
  - reset, 243
  - setBodyBytes, 243
  - toString, 244
  - writeBoolean, 244
  - writeByte, 244
  - writeBytes, 245
  - writeChar, 245
  - writeDouble, 246
  - writeFloat, 246
  - writeInt, 246
  - writeLong, 247
  - writeShort, 247
  - writeString, 247
  - writeUnsignedShort, 247
  - writeUTF, 248
- activemq::commands::ActiveMQDestination, 322
  - ~ActiveMQDestination, 324
  - ActiveMQDestination, 324
  - advisory, 331
  - ADVISORY\_PREFIX, 331
  - cloneDataStructure, 324
  - COMPOSITE\_SEPARATOR, 331
  - CONNECTION\_ADVISORY\_PREFIX, 331
  - CONSUMER\_ADVISORY\_PREFIX, 331



- copyDataStructure, 325
- createDestination, 325
- createTemporaryName, 325
- DEFAULT\_ORDERED\_TARGET, 331
- equals, 325
- exclusive, 331
- getClientId, 326
- getCMSDestination, 326
- getDataStructureType, 326
- getDestinationType, 327
- getOptions, 327
- getOrderedTarget, 327
- getPhysicalName, 327
- ID\_ACTIVEMQDESTINATION, 332
- isAdvisory, 327
- isComposite, 328
- isConnectionAdvisory, 328
- isConsumerAdvisory, 328
- isExclusive, 328
- isOrdered, 328
- isProducerAdvisory, 328
- isQueue, 329
- isTemporary, 329
- isTopic, 329
- isWildcard, 329
- options, 332
- ordered, 332
- orderedTarget, 332
- physicalName, 332
- PRODUCER\_ADVISORY\_PREFIX, 332
- QUEUE\_QUALIFIED\_PREFIX, 332
- setAdvisory, 329
- setExclusive, 329
- setOrdered, 330
- setOrderedTarget, 330
- setPhysicalName, 330
- TEMP\_POSTFIX, 332
- TEMP\_PREFIX, 332
- TEMP\_QUEUE\_QUALIFIED\_PREFIX, 332
- TEMP\_TOPIC\_QUALIFIED\_PREFIX, 332
- TOPIC\_QUALIFIED\_PREFIX, 332
- toString, 330
- activemq::commands::ActiveMQDestination::DestinationFilter, 1726
  - ANY\_CHILD, 1726
  - ANY\_DESCENDENT, 1726
- activemq::commands::ActiveMQMapMessage, 360
  - ~ActiveMQMapMessage, 363
  - ActiveMQMapMessage, 363
  - beforeMarshal, 363
  - checkMapIsUnmarshalled, 363
  - clearBody, 363
  - clone, 363
  - cloneDataStructure, 363
  - copyDataStructure, 364
  - equals, 364
  - getBoolean, 364
  - getByte, 364
  - getBytes, 365
  - getChar, 365
  - getDataStructureType, 365
  - getDouble, 366
  - getFloat, 366
  - getInt, 366
  - getLong, 367
  - getMap, 367
  - getMapNames, 367
  - getShort, 367
  - getString, 368
  - ID\_ACTIVEMQMAPMESSAGE, 372
  - isMarshalAware, 368
  - itemExists, 368
  - setBoolean, 369
  - setByte, 369
  - setBytes, 369
  - setChar, 370
  - setDouble, 370
  - setFloat, 370
  - setInt, 371
  - setLong, 371
  - setShort, 371
  - setString, 372
  - toString, 372
- activemq::commands::ActiveMQMessage, 398
  - ~ActiveMQMessage, 398
  - ActiveMQMessage, 398
  - clone, 398
  - cloneDataStructure, 399
  - copyDataStructure, 399
  - equals, 399
  - getDataStructureType, 399
  - ID\_ACTIVEMQMESSAGE, 400
  - toString, 399
- activemq::commands::ActiveMQMessageTemplate, 425
  - ~ActiveMQMessageTemplate, 428
  - ActiveMQMessageTemplate, 428
  - clearBody, 428
  - clearProperties, 429
  - equals, 429
  - failIfReadOnlyBody, 429
  - failIfReadOnlyProperties, 429
  - failIfWriteOnlyBody, 429
  - getBooleanProperty, 429

- getByteProperty, 430
- getCMSCorrelationID, 430
- getCMSDeliveryMode, 430
- getCMSDestination, 431
- getCMSExpiration, 431
- getCMSMessageID, 431
- getCMSPriority, 431
- getCMSRedelivered, 432
- getCMSReplyTo, 432
- getCMSTimestamp, 432
- getCMSType, 433
- getDoubleProperty, 433
- getFloatProperty, 433
- getIntProperty, 434
- getLongProperty, 434
- getPropertyNames, 434
- getShortProperty, 435
- getStringProperty, 435
- onSend, 435
- propertyExists, 436
- setBooleanProperty, 436
- setByteProperty, 436
- setCMSCorrelationID, 437
- setCMSDeliveryMode, 437
- setCMSDestination, 437
- setCMSExpiration, 437
- setCMSMessageID, 438
- setCMSPriority, 438
- setCMSRedelivered, 438
- setCMSReplyTo, 439
- setCMSTimestamp, 439
- setCMSType, 439
- setDoubleProperty, 439
- setFloatProperty, 440
- setIntProperty, 440
- setLongProperty, 440
- setShortProperty, 441
- setStringProperty, 441
- activemq::commands::ActiveMQObjectMessage, 443
  - ~ActiveMQObjectMessage, 444
  - ActiveMQObjectMessage, 444
  - clone, 444
  - cloneDataStructure, 444
  - copyDataStructure, 444
  - equals, 444
  - getDataStructureType, 445
  - ID\_ACTIVEMQOBJECTMESSAGE, 445
  - toString, 445
- activemq::commands::ActiveMQQueue, 483
  - ~ActiveMQQueue, 484
  - ActiveMQQueue, 484
  - clone, 484
  - cloneDataStructure, 484
  - copy, 484
  - copyDataStructure, 484
  - equals, 484
  - getCMSDestination, 485
  - getCMSProperties, 485
  - getDataStructureType, 485
  - getDestinationType, 485
  - getQueueName, 486
  - ID\_ACTIVEMQQUEUE, 486
  - toString, 486
- activemq::commands::ActiveMQStreamMessage, 537
  - ~ActiveMQStreamMessage, 540
  - ActiveMQStreamMessage, 540
  - clearBody, 540
  - clone, 540
  - cloneDataStructure, 540
  - copyDataStructure, 540
  - equals, 541
  - getDataStructureType, 541
  - ID\_ACTIVEMQSTREAMMESSAGE, 551
  - onSend, 541
  - readBoolean, 541
  - readByte, 542
  - readBytes, 542, 543
  - readChar, 543
  - readDouble, 544
  - readFloat, 544
  - readInt, 544
  - readLong, 545
  - readShort, 545
  - readString, 546
  - readUnsignedShort, 546
  - reset, 546
  - toString, 547
  - writeBoolean, 547
  - writeByte, 547
  - writeBytes, 548
  - writeChar, 548
  - writeDouble, 549
  - writeFloat, 549
  - writeInt, 549
  - writeLong, 549
  - writeShort, 550
  - writeString, 550
  - writeUnsignedShort, 550
- activemq::commands::ActiveMQTempDestination, 576
  - ~ActiveMQTempDestination, 577
  - ActiveMQTempDestination, 577
  - cloneDataStructure, 577
  - close, 577
  - connection, 579

- copyDataStructure, 577
- equals, 577
- getDataStructureType, 578
- ID\_ACTIVEMQTEMPDESTINATION, 579
- setConnection, 578
- toString, 578
- activemq::commands::ActiveMQTempQueue, 604
  - ~ActiveMQTempQueue, 605
  - ActiveMQTempQueue, 605
  - clone, 605
  - cloneDataStructure, 605
  - copy, 605
  - copyDataStructure, 605
  - destroy, 606
  - equals, 606
  - getCMSDestination, 606
  - getCMSProperties, 606
  - getDataStructureType, 607
  - getDestinationType, 607
  - getQueueName, 607
  - ID\_ACTIVEMQTEMPQUEUE, 608
  - toString, 607
- activemq::commands::ActiveMQTempTopic, 633
  - ~ActiveMQTempTopic, 634
  - ActiveMQTempTopic, 634
  - clone, 634
  - cloneDataStructure, 634
  - copy, 634
  - copyDataStructure, 634
  - destroy, 635
  - equals, 635
  - getCMSDestination, 635
  - getCMSProperties, 635
  - getDataStructureType, 636
  - getDestinationType, 636
  - getTopicName, 636
  - ID\_ACTIVEMQTEMPTOPIC, 637
  - toString, 636
- activemq::commands::ActiveMQTextMessage, 662
  - ~ActiveMQTextMessage, 663
  - ActiveMQTextMessage, 663
  - beforeMarshal, 663
  - clearBody, 663
  - clone, 663
  - cloneDataStructure, 664
  - copyDataStructure, 664
  - equals, 664
  - getDataStructureType, 664
  - getSize, 664
  - getText, 665
  - ID\_ACTIVEMQTEXTMESSAGE, 666
  - setText, 665
  - text, 666
  - toString, 666
- activemq::commands::ActiveMQTopic, 691
  - ~ActiveMQTopic, 692
  - ActiveMQTopic, 692
  - clone, 692
  - cloneDataStructure, 692
  - copy, 692
  - copyDataStructure, 692
  - equals, 692
  - getCMSDestination, 693
  - getCMSProperties, 693
  - getDataStructureType, 693
  - getDestinationType, 693
  - getTopicName, 694
  - ID\_ACTIVEMQTOPIC, 694
  - toString, 694
- activemq::commands::BaseCommand, 758
  - ~BaseCommand, 759
  - BaseCommand, 759
  - copyDataStructure, 759
  - equals, 759
  - getCommandId, 760
  - isBrokerInfo, 760
  - isConnectionInfo, 761
  - isConsumerInfo, 761
  - isKeepAliveInfo, 761
  - isMessage, 761
  - isMessageAck, 761
  - isMessageDispatch, 761
  - isMessageDispatchNotification, 761
  - isProducerAck, 761
  - isProducerInfo, 762
  - isRemoveInfo, 762
  - isRemoveSubscriptionInfo, 762
  - isResponse, 762
  - isResponseRequired, 762
  - isShutdownInfo, 762
  - isTransactionInfo, 762
  - isWireFormatInfo, 763
  - setCommandId, 763
  - setResponseRequired, 763
  - toString, 763
- activemq::commands::BaseDataStructure, 830
  - ~BaseDataStructure, 831
  - afterMarshal, 831
  - afterUnmarshal, 831
  - beforeMarshal, 831
  - beforeUnmarshal, 831
  - copyDataStructure, 831
  - equals, 832
  - getMarshaledForm, 832

- isMarshalAware, 832
- setMarshaledForm, 832
- toString, 833
- activemq::commands::BooleanExpression, 855
  - ~BooleanExpression, 855
  - BooleanExpression, 855
  - cloneDataStructure, 855
  - copyDataStructure, 855
  - equals, 856
  - toString, 856
- activemq::commands::BrokerError, 863
  - ~BrokerError, 864
  - BrokerError, 864
  - cloneDataStructure, 864
  - copyDataStructure, 864
  - getCause, 864
  - getDataStructureType, 865
  - getExceptionClass, 865
  - getMessage, 865
  - getStackTraceElements, 865
  - setCause, 865
  - setExceptionClass, 866
  - setMessage, 866
  - setStackTraceElements, 866
  - visit, 866
- activemq::commands::BrokerError::StackTraceElement, 3578
  - ClassName, 3578
  - FileName, 3578
  - LineNumber, 3578
  - MethodName, 3578
- activemq::commands::BrokerId, 869
  - ~BrokerId, 870
  - BrokerId, 870
  - cloneDataStructure, 870
  - COMPARATOR, 870
  - compareTo, 870
  - copyDataStructure, 870
  - equals, 870
  - getDataStructureType, 870
  - getValue, 871
  - ID\_BROKERID, 871
  - operator<, 871
  - operator=, 871
  - operator==, 871
  - setValue, 871
  - toString, 871
  - value, 871
- activemq::commands::BrokerInfo, 896
  - ~BrokerInfo, 897
  - brokerId, 903
  - BrokerInfo, 897
  - brokerName, 903
  - brokerUploadUrl, 903
  - brokerURL, 903
  - cloneDataStructure, 897
  - connectionId, 903
  - copyDataStructure, 898
  - duplexConnection, 903
  - equals, 898
  - faultTolerantConfiguration, 903
  - getBrokerId, 898, 899
  - getBrokerName, 899
  - getBrokerUploadUrl, 899
  - getBrokerURL, 899
  - getConnectionId, 899
  - getDataStructureType, 899
  - getNetworkProperties, 899, 900
  - getPeerBrokerInfos, 900
  - ID\_BROKERINFO, 903
  - isBrokerInfo, 900
  - isDuplexConnection, 900
  - isFaultTolerantConfiguration, 901
  - isMasterBroker, 901
  - isNetworkConnection, 901
  - isSlaveBroker, 901
  - masterBroker, 903
  - networkConnection, 903
  - networkProperties, 903
  - peerBrokerInfos, 903
  - setBrokerId, 901
  - setBrokerName, 901
  - setBrokerUploadUrl, 901
  - setBrokerURL, 901
  - setConnectionId, 901
  - setDuplexConnection, 901
  - setFaultTolerantConfiguration, 901
  - setMasterBroker, 901
  - setNetworkConnection, 901
  - setNetworkProperties, 901
  - setPeerBrokerInfos, 901
  - setSlaveBroker, 901
  - slaveBroker, 903
  - toString, 901
  - visit, 902
- activemq::commands::Command, 1194
  - ~Command, 1195
  - getCommandId, 1195
  - isBrokerInfo, 1195
  - isConnectionInfo, 1195
  - isConsumerInfo, 1195
  - isKeepAliveInfo, 1195
  - isMessage, 1195
  - isMessageAck, 1195
  - isMessageDispatch, 1196
  - isMessageDispatchNotification, 1196
  - isProducerAck, 1196
  - isProducerInfo, 1196

- isRemoveInfo, 1196
- isRemoveSubscriptionInfo, 1196
- isResponse, 1196
- isResponseRequired, 1196
- isShutdownInfo, 1197
- isTransactionInfo, 1197
- isWireFormatInfo, 1197
- setCommandId, 1197
- setResponseRequired, 1197
- toString, 1197
- visit, 1198
- activemq::commands::ConnectionControl, 1267
  - ~ConnectionControl, 1268
  - cloneDataStructure, 1268
  - close, 1271
  - connectedBrokers, 1271
  - ConnectionControl, 1268
  - copyDataStructure, 1268
  - equals, 1268
  - exit, 1271
  - faultTolerant, 1271
  - getConnectedBrokers, 1269
  - getDataStructureType, 1269
  - getReconnectTo, 1269, 1270
  - ID\_CONNECTIONCONTROL, 1271
  - isClose, 1270
  - isExit, 1270
  - isFaultTolerant, 1270
  - isRebalanceConnection, 1270
  - isResume, 1270
  - isSuspend, 1270
  - rebalanceConnection, 1271
  - reconnectTo, 1271
  - resume, 1271
  - setClose, 1270
  - setConnectedBrokers, 1270
  - setExit, 1270
  - setFaultTolerant, 1270
  - setRebalanceConnection, 1270
  - setReconnectTo, 1270
  - setResume, 1270
  - setSuspend, 1270
  - suspend, 1271
  - toString, 1270
  - visit, 1271
- activemq::commands::ConnectionError, 1296
  - ~ConnectionError, 1297
  - cloneDataStructure, 1297
  - ConnectionError, 1297
  - connectionId, 1299
  - copyDataStructure, 1297
  - equals, 1297
  - exception, 1299
  - getConnectionId, 1297, 1298
  - getDataStructureType, 1298
  - getException, 1298
  - ID\_CONNECTIONERROR, 1299
  - setConnectionId, 1298
  - setException, 1298
  - toString, 1298
  - visit, 1298
- activemq::commands::ConnectionId, 1327
  - ~ConnectionId, 1328
  - cloneDataStructure, 1328
  - COMPARATOR, 1328
  - compareTo, 1328
  - ConnectionId, 1328
  - copyDataStructure, 1328
  - equals, 1328, 1329
  - getDataStructureType, 1329
  - getValue, 1329
  - ID\_CONNECTIONID, 1330
  - operator<, 1329
  - operator=, 1329
  - operator==, 1329
  - setValue, 1329
  - toString, 1329
  - value, 1330
- activemq::commands::ConnectionInfo, 1355
  - ~ConnectionInfo, 1356
  - brokerMasterConnector, 1360
  - brokerPath, 1360
  - clientId, 1360
  - clientMaster, 1360
  - cloneDataStructure, 1356
  - connectionId, 1360
  - ConnectionInfo, 1356
  - copyDataStructure, 1356
  - createRemoveCommand, 1357
  - equals, 1357
  - faultTolerant, 1360
  - getBrokerPath, 1357
  - getClientId, 1357
  - getConnectionId, 1357
  - getDataStructureType, 1357
  - getPassword, 1358
  - getUserName, 1358
  - ID\_CONNECTIONINFO, 1360
  - isBrokerMasterConnector, 1358
  - isClientMaster, 1358
  - isConnectionInfo, 1358
  - isFaultTolerant, 1358
  - isManageable, 1359
  - manageable, 1360
  - password, 1360
  - setBrokerMasterConnector, 1359
  - setBrokerPath, 1359
  - setClientId, 1359

- setClientMaster, 1359
- setConnectionId, 1359
- setFaultTolerant, 1359
- setManageable, 1359
- setPassword, 1359
- setUserName, 1359
- toString, 1359
- userName, 1360
- visit, 1359
- activemq::commands::ConsumerControl, 1401
  - ~ConsumerControl, 1402
  - cloneDataStructure, 1402
  - close, 1405
  - ConsumerControl, 1402
  - consumerId, 1405
  - copyDataStructure, 1402
  - destination, 1405
  - equals, 1402
  - flush, 1405
  - getConsumerId, 1403
  - getDataStructureType, 1403
  - getDestination, 1403, 1404
  - getPrefetch, 1404
  - ID\_CONSUMERCONTROL, 1405
  - isClose, 1404
  - isFlush, 1404
  - isStart, 1404
  - isStop, 1404
  - prefetch, 1405
  - setClose, 1404
  - setConsumerId, 1404
  - setDestination, 1404
  - setFlush, 1404
  - setPrefetch, 1404
  - setStart, 1404
  - setStop, 1404
  - start, 1405
  - stop, 1405
  - toString, 1404
  - visit, 1404
- activemq::commands::ConsumerId, 1430
  - ~ConsumerId, 1431
  - cloneDataStructure, 1431
  - COMPARATOR, 1431
  - compareTo, 1431
  - connectionId, 1433
  - ConsumerId, 1431
  - copyDataStructure, 1431
  - equals, 1431, 1432
  - getConnectionId, 1432
  - getDataStructureType, 1432
  - getParentId, 1432
  - getSessionId, 1433
  - getValue, 1433
  - ID\_CONSUMERID, 1433
  - operator<, 1433
  - operator=, 1433
  - operator==, 1433
  - sessionId, 1433
  - setConnectionId, 1433
  - setSessionId, 1433
  - setValue, 1433
  - toString, 1433
  - value, 1434
- activemq::commands::ConsumerInfo, 1459
  - ~ConsumerInfo, 1461
  - additionalPredicate, 1466
  - brokerPath, 1466
  - browser, 1466
  - cloneDataStructure, 1461
  - consumerId, 1466
  - ConsumerInfo, 1461
  - copyDataStructure, 1461
  - createRemoveCommand, 1461
  - destination, 1466
  - dispatchAsync, 1466
  - equals, 1461
  - exclusive, 1466
  - getAdditionalPredicate, 1462
  - getBrokerPath, 1462
  - getConsumerId, 1462
  - getDataStructureType, 1462
  - getDestination, 1462, 1463
  - getMaximumPendingMessageLimit, 1463
  - getNetworkConsumerPath, 1463
  - getPrefetchSize, 1463
  - getPriority, 1463
  - getSelector, 1463
  - getSubscriptionName, 1463
  - ID\_CONSUMERINFO, 1466
  - isBrowser, 1463
  - isConsumerInfo, 1463
  - isDispatchAsync, 1463
  - isExclusive, 1464
  - isNetworkSubscription, 1464
  - isNoLocal, 1464
  - isNoRangeAcks, 1464
  - isOptimizedAcknowledge, 1464
  - isRetroactive, 1464
  - maximumPendingMessageLimit, 1466
  - networkConsumerPath, 1466
  - networkSubscription, 1466
  - noLocal, 1466
  - noRangeAcks, 1466
  - optimizedAcknowledge, 1466
  - prefetchSize, 1466
  - priority, 1466
  - retroactive, 1466

- selector, 1466
- setAdditionalPredicate, 1464
- setBrokerPath, 1464
- setBrowser, 1464
- setConsumerId, 1464
- setDestination, 1464
- setDispatchAsync, 1464
- setExclusive, 1464
- setMaximumPendingMessageLimit, 1464
- setNetworkConsumerPath, 1464
- setNetworkSubscription, 1464
- setNoLocal, 1464
- setNoRangeAcks, 1464
- setOptimizedAcknowledge, 1464
- setPrefetchSize, 1464
- setPriority, 1464
- setRetroactive, 1464
- setSelector, 1464
- setSubscriptionName, 1464
- subscriptionName, 1466
- toString, 1464
- visit, 1465
- activemq::commands::ControlCommand, 1493
  - ~ControlCommand, 1494
  - cloneDataStructure, 1494
  - command, 1496
  - ControlCommand, 1494
  - copyDataStructure, 1494
  - equals, 1494
  - getCommand, 1494, 1495
  - getDataStructureType, 1495
  - ID\_CONTROLCOMMAND, 1496
  - setCommand, 1495
  - toString, 1495
  - visit, 1495
- activemq::commands::DataArrayResponse, 1528
  - ~DataArrayResponse, 1529
  - cloneDataStructure, 1529
  - copyDataStructure, 1529
  - data, 1530
  - DataArrayResponse, 1529
  - equals, 1529
  - getData, 1529, 1530
  - getDataStructureType, 1530
  - ID\_DATAARRAYRESPONSE, 1530
  - setData, 1530
  - toString, 1530
- activemq::commands::DataResponse, 1583
  - ~DataResponse, 1584
  - cloneDataStructure, 1584
  - copyDataStructure, 1584
  - data, 1585
  - DataResponse, 1584
  - equals, 1584
  - getData, 1584, 1585
  - getDataStructureType, 1585
  - ID\_DATARESPONSE, 1585
  - setData, 1585
  - toString, 1585
- activemq::commands::DataStructure, 1660
  - ~DataStructure, 1660
  - cloneDataStructure, 1660
  - copyDataStructure, 1661
  - equals, 1661
  - getDataStructureType, 1662
  - toString, 1663
- activemq::commands::DestinationInfo, 1727
  - ~DestinationInfo, 1728
  - brokerPath, 1731
  - cloneDataStructure, 1728
  - connectionId, 1731
  - copyDataStructure, 1728
  - destination, 1731
  - DestinationInfo, 1728
  - equals, 1728
  - getBrokerPath, 1729
  - getConnectionId, 1729
  - getDataStructureType, 1729
  - getDestination, 1729, 1730
  - getOperationType, 1730
  - getTimeout, 1730
  - ID\_DESTINATIONINFO, 1731
  - operationType, 1731
  - setBrokerPath, 1730
  - setConnectionId, 1730
  - setDestination, 1730
  - setOperationType, 1730
  - setTimeout, 1730
  - timeout, 1731
  - toString, 1730
  - visit, 1730
- activemq::commands::DiscoveryEvent, 1758
  - ~DiscoveryEvent, 1759
  - brokerName, 1761
  - cloneDataStructure, 1759
  - copyDataStructure, 1759
  - DiscoveryEvent, 1759
  - equals, 1759
  - getBrokerName, 1759, 1760
  - getDataStructureType, 1760
  - getServiceName, 1760
  - ID\_DISCOVERYEVENT, 1761
  - serviceName, 1761
  - setBrokerName, 1760
  - setServiceName, 1760
  - toString, 1760
- activemq::commands::ExceptionResponse, 1839

- ~ExceptionResponse, 1840
- cloneDataStructure, 1840
- copyDataStructure, 1840
- equals, 1840
- exception, 1841
- ExceptionResponse, 1840
- getDataStructureType, 1840
- getException, 1841
- ID\_EXCEPTIONRESPONSE, 1841
- setException, 1841
- toString, 1841
- activemq::commands::FlushCommand, 1937
  - ~FlushCommand, 1938
  - cloneDataStructure, 1938
  - copyDataStructure, 1938
  - equals, 1938
  - FlushCommand, 1938
  - getDataStructureType, 1938
  - ID\_FLUSHCOMMAND, 1939
  - toString, 1939
  - visit, 1939
- activemq::commands::IntegerResponse, 2091
  - ~IntegerResponse, 2092
  - cloneDataStructure, 2092
  - copyDataStructure, 2092
  - equals, 2092
  - getDataStructureType, 2092
  - getResult, 2093
  - ID\_INTEGERRESPONSE, 2093
  - IntegerResponse, 2092
  - result, 2093
  - setResult, 2093
  - toString, 2093
- activemq::commands::JournalQueueAck, 2156
  - ~JournalQueueAck, 2157
  - cloneDataStructure, 2157
  - copyDataStructure, 2157
  - destination, 2158
  - equals, 2157
  - getDataStructureType, 2157
  - getDestination, 2157, 2158
  - getMessageAck, 2158
  - ID\_JOURNALQUEUEACK, 2158
  - JournalQueueAck, 2157
  - messageAck, 2158
  - setDestination, 2158
  - setMessageAck, 2158
  - toString, 2158
- activemq::commands::JournalTopicAck, 2183
  - ~JournalTopicAck, 2184
  - clientId, 2187
  - cloneDataStructure, 2184
  - copyDataStructure, 2184
  - destination, 2187
  - equals, 2184
  - getClientId, 2184, 2185
  - getDataStructureType, 2185
  - getDestination, 2185, 2186
  - getMessageId, 2186
  - getMessageSequenceId, 2186
  - getSubscriptionName, 2186
  - getTransactionId, 2186
  - ID\_JOURNALTOPICACK, 2187
  - JournalTopicAck, 2184
  - messageId, 2187
  - messageSequenceId, 2187
  - setClientId, 2186
  - setDestination, 2186
  - setMessageId, 2186
  - setMessageSequenceId, 2186
  - setSubscriptionName, 2186
  - setTransactionId, 2186
  - subscriptionName, 2187
  - toString, 2186
  - transactionId, 2187
- activemq::commands::JournalTrace, 2212
  - ~JournalTrace, 2213
  - cloneDataStructure, 2213
  - copyDataStructure, 2213
  - equals, 2213
  - getDataStructureType, 2213
  - getMessage, 2213, 2214
  - ID\_JOURNALTRACE, 2214
  - JournalTrace, 2213
  - message, 2214
  - setMessage, 2214
  - toString, 2214
- activemq::commands::JournalTransaction, 2239
  - ~JournalTransaction, 2240
  - cloneDataStructure, 2240
  - copyDataStructure, 2240
  - equals, 2240
  - getDataStructureType, 2240
  - getTransactionId, 2240, 2241
  - getType, 2241
  - getWasPrepared, 2241
  - ID\_JOURNALTRANSACTION, 2241
  - JournalTransaction, 2240
  - setTransactionId, 2241
  - setType, 2241
  - setWasPrepared, 2241
  - toString, 2241
  - transactionId, 2241
  - type, 2241
  - wasPrepared, 2241
- activemq::commands::KeepAliveInfo, 2266
  - ~KeepAliveInfo, 2267
  - cloneDataStructure, 2267



- copyDataStructure, 2267
- equals, 2267
- getDataStructureType, 2267
- ID\_KEEPLIVEINFO, 2268
- isKeepAliveInfo, 2268
- KeepAliveInfo, 2267
- toString, 2268
- visit, 2268
- activemq::commands::LastPartialCommand, 2301
  - ~LastPartialCommand, 2301
  - cloneDataStructure, 2301
  - copyDataStructure, 2302
  - equals, 2302
  - getDataStructureType, 2302
  - ID\_LASTPARTIALCOMMAND, 2303
  - LastPartialCommand, 2301
  - toString, 2302
- activemq::commands::LocalTransactionId, 2347
  - ~LocalTransactionId, 2348
  - cloneDataStructure, 2348
  - COMPARATOR, 2348
  - compareTo, 2348
  - connectionId, 2350
  - copyDataStructure, 2348
  - equals, 2348, 2349
  - getConnectionId, 2349
  - getDataStructureType, 2349
  - getValue, 2349
  - ID\_LOCALTRANSACTIONID, 2350
  - LocalTransactionId, 2348
  - operator<, 2349
  - operator=, 2349
  - operator==, 2349
  - setConnectionId, 2350
  - setValue, 2350
  - toString, 2350
  - value, 2350
- activemq::commands::Message, 2516
  - ~Message, 2520
  - afterUnmarshal, 2520
  - arrival, 2532
  - beforeMarshal, 2520
  - brokerInTime, 2532
  - brokerOutTime, 2532
  - brokerPath, 2532
  - cloneDataStructure, 2520
  - cluster, 2532
  - compressed, 2532
  - connection, 2532
  - content, 2532
  - copyDataStructure, 2521
  - correlationId, 2532
  - dataStructure, 2532
  - DEFAULT\_MESSAGE\_SIZE, 2532
  - destination, 2532
  - droppable, 2532
  - equals, 2521
  - expiration, 2532
  - getAckHandler, 2521
  - getArrival, 2522
  - getBrokerInTime, 2522
  - getBrokerOutTime, 2522
  - getBrokerPath, 2522
  - getCluster, 2522
  - getConnection, 2522
  - getContent, 2522, 2523
  - getCorrelationId, 2523
  - getDataStructure, 2523
  - getDataStructureType, 2523
  - getDestination, 2523, 2524
  - getExpiration, 2524
  - getGroupID, 2524
  - getGroupSequence, 2524
  - getMarshaledProperties, 2524
  - getMessageId, 2524
  - getMessageProperties, 2524
  - getOriginalDestination, 2524, 2525
  - getOriginalTransactionId, 2525
  - getPriority, 2525
  - getProducerId, 2525
  - getRedeliveryCounter, 2525
  - getReplyTo, 2525
  - getSize, 2525
  - getTargetConsumerId, 2525, 2526
  - getTimestamp, 2526
  - getTransactionId, 2526
  - getType, 2526
  - getUserID, 2526
  - groupID, 2532
  - groupSequence, 2532
  - ID\_MESSAGE, 2532
  - isCompressed, 2526
  - isDroppable, 2526
  - isExpired, 2526
  - isMarshalAware, 2526
  - isMessage, 2527
  - isPersistent, 2527
  - isReadOnlyBody, 2527
  - isReadOnlyProperties, 2527
  - isRecievedByDFBridge, 2527
  - marshalledProperties, 2532
  - Message, 2520
  - messageId, 2532
  - onSend, 2527
  - originalDestination, 2532
  - originalTransactionId, 2532
  - persistent, 2532

- priority, 2532
- producerId, 2532
- recievedByDFBridge, 2532
- redeliveryCounter, 2532
- replyTo, 2532
- setAckHandler, 2527
- setArrival, 2528
- setBrokerInTime, 2528
- setBrokerOutTime, 2528
- setBrokerPath, 2528
- setCluster, 2528
- setCompressed, 2528
- setConnection, 2528
- setContent, 2528
- setCorrelationId, 2529
- setDataStructure, 2529
- setDestination, 2529
- setDroppable, 2529
- setExpiration, 2529
- setGroupID, 2529
- setGroupSequence, 2529
- setMarshaledProperties, 2529
- setMessageId, 2529
- setOriginalDestination, 2529
- setOriginalTransactionId, 2529
- setPersistent, 2529
- setPriority, 2529
- setProducerId, 2529
- setReadOnlyBody, 2529
- setReadOnlyProperties, 2529
- setRecievedByDFBridge, 2530
- setRedeliveryCounter, 2530
- setReplyTo, 2530
- setTargetConsumerId, 2530
- setTimestamp, 2530
- setTransactionId, 2530
- setType, 2530
- setUserID, 2530
- targetConsumerId, 2532
- timestamp, 2532
- toString, 2530
- transactionId, 2532
- type, 2532
- userId, 2532
- visit, 2530
- activemq::commands::MessageAck, 2559
  - ~MessageAck, 2560
  - ackType, 2564
  - cloneDataStructure, 2560
  - consumerId, 2564
  - copyDataStructure, 2560
  - destination, 2564
  - equals, 2560
  - firstMessageId, 2564
  - getAckType, 2561
  - getConsumerId, 2561
  - getDataStructureType, 2561
  - getDestination, 2561, 2562
  - getFirstMessageId, 2562
  - getLastMessageId, 2562
  - getMessageCount, 2562
  - getTransactionId, 2562
  - ID\_MESSAGEACK, 2564
  - isMessageAck, 2562
  - lastMessageId, 2564
  - MessageAck, 2560
  - messageCount, 2564
  - setAckType, 2562
  - setConsumerId, 2563
  - setDestination, 2563
  - setFirstMessageId, 2563
  - setLastMessageId, 2563
  - setMessageCount, 2563
  - setTransactionId, 2563
  - toString, 2563
  - transactionId, 2564
  - visit, 2563
- activemq::commands::MessageDispatch, 2594
  - ~MessageDispatch, 2595
  - cloneDataStructure, 2595
  - consumerId, 2598
  - copyDataStructure, 2595
  - destination, 2598
  - equals, 2595
  - getConsumerId, 2595, 2596
  - getDataStructureType, 2596
  - getDestination, 2596
  - getMessage, 2596
  - getRedeliveryCounter, 2596
  - ID\_MESSAGEDISPATCH, 2598
  - isMessageDispatch, 2596
  - message, 2598
  - MessageDispatch, 2595
  - redeliveryCounter, 2598
  - setConsumerId, 2596
  - setDestination, 2597
  - setMessage, 2597
  - setRedeliveryCounter, 2597
  - toString, 2597
  - visit, 2597
- activemq::commands::MessageDispatchNotification, 2630
  - ~MessageDispatchNotification, 2631
  - cloneDataStructure, 2631
  - consumerId, 2634
  - copyDataStructure, 2631
  - deliverySequenceId, 2634
  - destination, 2634

- equals, 2631
- getConsumerId, 2631, 2632
- getDataStructureType, 2632
- getDeliverySequenceId, 2632
- getDestination, 2632
- getMessageId, 2632
- ID\_MESSAGEDISPATCHNOTIFICATION, 2634
- isMessageDispatchNotification, 2632
- MessageDispatchNotification, 2631
- messageId, 2634
- setConsumerId, 2632
- setDeliverySequenceId, 2633
- setDestination, 2633
- setMessageId, 2633
- toString, 2633
- visit, 2633
- activemq::commands::MessageId, 2663
  - ~MessageId, 2664
  - brokerSequenceId, 2667
  - cloneDataStructure, 2664
  - COMPARATOR, 2664
  - compareTo, 2664
  - copyDataStructure, 2665
  - equals, 2665
  - getBrokerSequenceId, 2665
  - getDataStructureType, 2665
  - getProducerId, 2665, 2666
  - getProducerSequenceId, 2666
  - ID\_MESSAGEID, 2667
  - MessageId, 2664
  - operator<, 2666
  - operator=, 2666
  - operator==, 2666
  - producerId, 2667
  - producerSequenceId, 2667
  - setBrokerSequenceId, 2666
  - setProducerId, 2666
  - setProducerSequenceId, 2666
  - setTextView, 2666
  - setValue, 2666
  - toString, 2666
- activemq::commands::MessagePull, 2739
  - ~MessagePull, 2740
  - cloneDataStructure, 2740
  - consumerId, 2743
  - copyDataStructure, 2740
  - correlationId, 2743
  - destination, 2743
  - equals, 2740
  - getConsumerId, 2740, 2741
  - getCorrelationId, 2741
  - getDataStructureType, 2741
  - getDestination, 2741, 2742
  - getMessageId, 2742
  - getTimeout, 2742
  - ID\_MESSAGEPULL, 2743
  - messageId, 2743
  - MessagePull, 2740
  - setConsumerId, 2742
  - setCorrelationId, 2742
  - setDestination, 2742
  - setMessageId, 2742
  - setTimeout, 2742
  - timeout, 2743
  - toString, 2742
  - visit, 2742
- activemq::commands::NetworkBridgeFilter, 2793
  - ~NetworkBridgeFilter, 2794
  - cloneDataStructure, 2794
  - copyDataStructure, 2794
  - equals, 2794
  - getDataStructureType, 2794
  - getNetworkBrokerId, 2794, 2795
  - getNetworkTTL, 2795
  - ID\_NETWORKBRIDGEFILTER, 2795
  - NetworkBridgeFilter, 2794
  - networkBrokerId, 2795
  - networkTTL, 2795
  - setNetworkBrokerId, 2795
  - setNetworkTTL, 2795
  - toString, 2795
- activemq::commands::PartialCommand, 2918
  - ~PartialCommand, 2919
  - cloneDataStructure, 2919
  - commandId, 2921
  - copyDataStructure, 2919
  - data, 2921
  - equals, 2919
  - getCommandId, 2919
  - getData, 2920
  - getDataStructureType, 2920
  - ID\_PARTIALCOMMAND, 2921
  - PartialCommand, 2919
  - setCommandId, 2920
  - setData, 2920
  - toString, 2920
- activemq::commands::ProducerAck, 3036
  - ~ProducerAck, 3037
  - cloneDataStructure, 3037
  - copyDataStructure, 3037
  - equals, 3037
  - getDataStructureType, 3037
  - getProducerId, 3037, 3038
  - getSize, 3038
  - ID\_PRODUCERACK, 3039
  - isProducerAck, 3038

- ProducerAck, 3037
- producerId, 3039
- setProducerId, 3038
- setSize, 3038
- size, 3039
- toString, 3038
- visit, 3038
- activemq::commands::ProducerId, 3067
  - ~ProducerId, 3068
  - cloneDataStructure, 3068
  - COMPARATOR, 3068
  - compareTo, 3068
  - connectionId, 3070
  - copyDataStructure, 3068
  - equals, 3069
  - getConnectionId, 3069
  - getDataStructureType, 3069
  - getParentId, 3069
  - getSessionId, 3070
  - getValue, 3070
  - ID\_PRODUCERID, 3070
  - operator<, 3070
  - operator=, 3070
  - operator==, 3070
  - ProducerId, 3068
  - sessionId, 3070
  - setConnectionId, 3070
  - setProducerSessionKey, 3070
  - setSessionId, 3070
  - setValue, 3070
  - toString, 3070
  - value, 3071
- activemq::commands::ProducerInfo, 3096
  - ~ProducerInfo, 3097
  - brokerPath, 3100
  - cloneDataStructure, 3097
  - copyDataStructure, 3097
  - createRemoveCommand, 3097
  - destination, 3100
  - dispatchAsync, 3100
  - equals, 3097
  - getBrokerPath, 3098
  - getDataStructureType, 3098
  - getDestination, 3098
  - getProducerId, 3098
  - getWindowSize, 3098
  - ID\_PRODUCERINFO, 3100
  - isDispatchAsync, 3098
  - isProducerInfo, 3098
  - producerId, 3100
  - ProducerInfo, 3097
  - setBrokerPath, 3098
  - setDestination, 3099
  - setDispatchAsync, 3099
  - setProducerId, 3099
  - setWindowSize, 3099
  - toString, 3099
  - visit, 3099
  - windowSize, 3100
- activemq::commands::RemoveInfo, 3194
  - ~RemoveInfo, 3195
  - cloneDataStructure, 3195
  - copyDataStructure, 3195
  - equals, 3195
  - getDataStructureType, 3195
  - getLastDeliveredSequenceId, 3195
  - getObjectId, 3196
  - ID\_REMOVEINFO, 3197
  - isRemoveInfo, 3196
  - lastDeliveredSequenceId, 3197
  - objectId, 3197
  - RemoveInfo, 3195
  - setLastDeliveredSequenceId, 3196
  - setObjectId, 3196
  - toString, 3196
  - visit, 3196
- activemq::commands::RemoveSubscriptionInfo, 3222
  - ~RemoveSubscriptionInfo, 3223
  - clientId, 3226
  - cloneDataStructure, 3223
  - connectionId, 3226
  - copyDataStructure, 3223
  - equals, 3223
  - getClientId, 3223, 3224
  - getConnectionId, 3224
  - getDataStructureType, 3224
  - getSubscriptionName, 3224
  - ID\_REMOVESUBSCRIPTIONINFO, 3226
  - isRemoveSubscriptionInfo, 3224
  - RemoveSubscriptionInfo, 3223
  - setClientId, 3224
  - setConnectionId, 3225
  - setSubscriptionName, 3225
  - subscriptionName, 3226
  - toString, 3225
  - visit, 3225
- activemq::commands::ReplayCommand, 3251
  - ~ReplayCommand, 3252
  - cloneDataStructure, 3252
  - copyDataStructure, 3252
  - equals, 3252
  - firstNakNumber, 3253
  - getDataStructureType, 3252
  - getFirstNakNumber, 3253
  - getLastNakNumber, 3253
  - ID\_REPLAYCOMMAND, 3253

- lastNakNumber, 3253
- ReplayCommand, 3252
- setFirstNakNumber, 3253
- setLastNakNumber, 3253
- toString, 3253
- visit, 3253
- activemq::commands::Response, 3285
  - ~Response, 3286
  - cloneDataStructure, 3286
  - copyDataStructure, 3286
  - correlationId, 3288
  - equals, 3286
  - getCorrelationId, 3286
  - getDataStructureType, 3287
  - ID\_RESPONSE, 3288
  - isResponse, 3287
  - Response, 3286
  - setCorrelationId, 3287
  - toString, 3287
  - visit, 3287
- activemq::commands::SessionId, 3379
  - ~SessionId, 3380
  - cloneDataStructure, 3380
  - COMPARATOR, 3380
  - compareTo, 3380
  - connectionId, 3382
  - copyDataStructure, 3380
  - equals, 3381
  - getConnectionId, 3381
  - getDataStructureType, 3381
  - getParentId, 3381
  - getValue, 3382
  - ID\_SESSIONID, 3382
  - operator<, 3382
  - operator=, 3382
  - operator==, 3382
  - SessionId, 3380
  - setConnectionId, 3382
  - setValue, 3382
  - toString, 3382
  - value, 3382
- activemq::commands::SessionInfo, 3407
  - ~SessionInfo, 3408
  - cloneDataStructure, 3408
  - copyDataStructure, 3408
  - createRemoveCommand, 3408
  - equals, 3408
  - getAckMode, 3408
  - getDataStructureType, 3409
  - getSessionId, 3409
  - ID\_SESSIONINFO, 3410
  - sessionId, 3410
  - SessionInfo, 3408
  - setAckMode, 3409
  - setSessionId, 3409
  - toString, 3409
  - visit, 3409
- activemq::commands::ShutdownInfo, 3470
  - ~ShutdownInfo, 3471
  - cloneDataStructure, 3471
  - copyDataStructure, 3471
  - equals, 3471
  - getDataStructureType, 3471
  - ID\_SHUTDOWNINFO, 3472
  - isShutdownInfo, 3472
  - ShutdownInfo, 3471
  - toString, 3472
  - visit, 3472
- activemq::commands::SubscriptionInfo, 3671
  - ~SubscriptionInfo, 3672
  - clientId, 3675
  - cloneDataStructure, 3672
  - copyDataStructure, 3672
  - destination, 3675
  - equals, 3672
  - getClientId, 3672, 3673
  - getDataStructureType, 3673
  - getDestination, 3673, 3674
  - getSelector, 3674
  - getSubscriptionName, 3674
  - getSubscribedDestination, 3674
  - ID\_SUBSCRIPTIONINFO, 3675
  - selector, 3675
  - setClientId, 3674
  - setDestination, 3674
  - setSelector, 3674
  - setSubscriptionName, 3674
  - setSubscribedDestination, 3674
  - subscriptionName, 3675
  - subscribedDestination, 3675
  - SubscriptionInfo, 3672
  - toString, 3674
- activemq::commands::TransactionId, 3819
  - ~TransactionId, 3820
  - cloneDataStructure, 3820
  - COMPARATOR, 3819
  - compareTo, 3820
  - copyDataStructure, 3820
  - equals, 3820
  - getDataStructureType, 3821
  - ID\_TRANSACTIONID, 3822
  - operator<, 3821
  - operator=, 3821
  - operator==, 3821
  - toString, 3821
  - TransactionId, 3820
- activemq::commands::TransactionInfo, 3847
  - ~TransactionInfo, 3848

- cloneDataStructure, 3848
- connectionId, 3850
- copyDataStructure, 3848
- equals, 3848
- getConnectionId, 3848, 3849
- getDataStructureType, 3849
- getTransactionId, 3849
- getType, 3849
- ID\_TRANSACTIONINFO, 3850
- isTransactionInfo, 3849
- setConnectionId, 3849
- setTransactionId, 3849
- setType, 3849
- toString, 3849
- transactionId, 3850
- TransactionInfo, 3848
- type, 3850
- visit, 3850
- activemq::commands::WireFormatInfo, 3982
  - ~WireFormatInfo, 3984
  - afterUnmarshal, 3984
  - beforeMarshal, 3985
  - cloneDataStructure, 3985
  - copyDataStructure, 3985
  - equals, 3985
  - getCacheSize, 3985
  - getDataStructureType, 3986
  - getMagic, 3986
  - getMarshaledProperties, 3986
  - getMaxInactivityDuration, 3986
  - getMaxInactivityDurationInitialDelay, 3986
  - getProperties, 3987
  - getVersion, 3987
  - ID\_WIREFORMATINFO, 3991
  - isCacheEnabled, 3987
  - isMarshalAware, 3987
  - isSizePrefixDisabled, 3988
  - isStackTraceEnabled, 3988
  - isTcpNoDelayEnabled, 3988
  - isTightEncodingEnabled, 3988
  - isValid, 3988
  - isWireFormatInfo, 3988
  - setCacheEnabled, 3989
  - setCacheSize, 3989
  - setMagic, 3989
  - setMarshaledProperties, 3989
  - setMaxInactivityDuration, 3989
  - setMaxInactivityDurationInitialDelay, 3989
  - setProperties, 3990
  - setSizePrefixDisabled, 3990
  - setStackTraceEnabled, 3990
  - setTcpNoDelayEnabled, 3990
  - setTightEncodingEnabled, 3990
  - setVersion, 3990
  - toString, 3991
  - visit, 3991
  - WireFormatInfo, 3984
- activemq::commands::XATransactionId, 4031
  - ~XATransactionId, 4032
  - branchQualifier, 4035
  - cloneDataStructure, 4032
  - COMPARATOR, 4032
  - compareTo, 4032
  - copyDataStructure, 4032
  - equals, 4033
  - formatId, 4035
  - getBranchQualifier, 4033
  - getDataStructureType, 4033
  - getFormatId, 4033
  - getGlobalTransactionId, 4034
  - globalTransactionId, 4035
  - ID\_XATRANSACTIONID, 4035
  - operator<, 4034
  - operator=, 4034
  - operator==, 4034
  - setBranchQualifier, 4034
  - setFormatId, 4034
  - setGlobalTransactionId, 4034
  - toString, 4034
  - XATransactionId, 4032
- activemq::core, 89
- activemq::core::ActiveMQAckHandler, 203
  - ~ActiveMQAckHandler, 203
  - acknowledgeMessage, 203
- activemq::core::ActiveMQConnection, 273
  - ~ActiveMQConnection, 278
  - ActiveMQConnection, 278
  - addDispatcher, 278
  - addProducer, 278
  - addTransportListener, 278
  - close, 279
  - createSession, 279
  - destroyDestination, 279, 280
  - fire, 280
  - getBrokerURL, 280
  - getClientID, 280
  - getCloseTimeout, 281
  - getConnectionId, 281
  - getConnectionInfo, 281
  - getExceptionListener, 281
  - getMetaData, 281
  - getNextLocalTransactionId, 282
  - getNextSessionId, 282
  - getNextTempDestinationId, 282
  - getPassword, 282
  - getPrefetchPolicy, 283
  - getProducerWindowSize, 283
  - getRedeliveryPolicy, 283

- getSendTimeout, 283
- getTransport, 283
- getUsername, 284
- isAlwaysSyncSend, 284
- isClosed, 284
- isDispatchAsync, 284
- isStarted, 284
- isTransportFailed, 284
- isUseAsyncSend, 285
- isUseCompression, 285
- onCommand, 285
- oneway, 285
- onException, 285
- removeDispatcher, 286
- removeProducer, 286
- removeSession, 286
- removeTransportListener, 286
- sendPullRequest, 286
- setAlwaysSyncSend, 287
- setBrokerURL, 287
- setClientID, 287
- setCloseTimeout, 288
- setDefaultClientId, 288
- setDispatchAsync, 288
- setExceptionHandler, 288
- setPassword, 288
- setPrefetchPolicy, 288
- setProducerWindowSize, 289
- setRedeliveryPolicy, 289
- setSendTimeout, 289
- setTransportInterruptionProcessingComplete, 289
- setUseAsyncSend, 289
- setUseCompression, 290
- setUsername, 290
- start, 290
- stop, 290
- syncRequest, 290
- transportInterrupted, 291
- transportResumed, 291
- activemq::core::ActiveMQConnectionFactory, 292
  - ~ActiveMQConnectionFactory, 294
  - ActiveMQConnectionFactory, 294
  - createConnection, 295, 296
  - DEFAULT\_URI, 302
  - getBrokerURL, 296
  - getClientId, 296
  - getCloseTimeout, 297
  - getExceptionHandler, 297
  - getPassword, 297
  - getPrefetchPolicy, 297
  - getProducerWindowSize, 297
  - getRedeliveryPolicy, 298
  - getSendTimeout, 298
  - getUsername, 298
  - isAlwaysSyncSend, 298
  - isDispatchAsync, 298
  - isUseAsyncSend, 299
  - isUseCompression, 299
  - setAlwaysSyncSend, 299
  - setBrokerURL, 299
  - setClientId, 299
  - setCloseTimeout, 299
  - setDispatchAsync, 300
  - setExceptionHandler, 300
  - setPassword, 300
  - setPrefetchPolicy, 300
  - setProducerWindowSize, 300
  - setRedeliveryPolicy, 301
  - setSendTimeout, 301
  - setUseAsyncSend, 301
  - setUseCompression, 301
  - setUsername, 301
- activemq::core::ActiveMQConnectionMetaData, 303
  - ~ActiveMQConnectionMetaData, 304
  - ActiveMQConnectionMetaData, 304
  - getCMSMajorVersion, 304
  - getCMSMinorVersion, 304
  - getCMSProviderName, 304
  - getCMSVersion, 305
  - getCMSXPathPropertyNames, 305
  - getProviderMajorVersion, 305
  - getProviderMinorVersion, 305
  - getProviderVersion, 306
- activemq::core::ActiveMQConstants, 307
  - ACK\_TYPE\_CONSUMED, 308
  - ACK\_TYPE\_DELIVERED, 308
  - ACK\_TYPE\_INDIVIDUAL, 308
  - ACK\_TYPE\_POISON, 308
  - ACK\_TYPE\_REDELIVERED, 308
  - AckType, 308
  - CONNECTION\_ALWAYS\_SYNC\_SEND, 309
  - CONNECTION\_CLOSE\_TIMEOUT, 309
  - CONNECTION\_DISPATCH\_ASYNC, 309
  - CONNECTION\_PRODUCER\_WINDOW\_SIZE, 309
  - CONNECTION\_SEND\_TIMEOUT, 309
  - CONNECTION\_USE\_ASYNC\_SEND, 309
  - CONNECTION\_USE\_COMPRESSION, 309
  - CONSUMER\_DISPATCH\_ASYNC, 308
  - CONSUMER\_EXCLUSIVE, 308
  - CONSUMER\_NOLOCAL, 308
  - CONSUMER\_PREFETCH\_SIZE, 308
  - CONSUMER\_PRIORITY, 308

- CONSUMER\_RETROACTIVE, 308
- CONSUMER\_SELECTOR, 308
- CUNSUMER\_-
  - MAXPENDINGMSGSLIMIT, 308
- DESTINATION\_ADD\_OPERATION, 308
- DESTINATION\_REMOVE\_OPERATION, 308
- DestinationActions, 308
- DestinationOption, 308
- NUM\_OPTIONS, 308
- NUM\_PARAMS, 309
- PARAM\_CLIENTID, 309
- PARAM\_PASSWORD, 309
- PARAM\_USERNAME, 309
- toDestinationOption, 309
- toString, 309
- toURIOption, 309
- TRANSACTION\_STATE\_BEGIN, 309
- TRANSACTION\_STATE\_-
  - COMMITONEPHASE, 309
- TRANSACTION\_STATE\_-
  - COMMITTWOPHASE, 309
- TRANSACTION\_STATE\_END, 309
- TRANSACTION\_STATE\_FORGET, 309
- TRANSACTION\_STATE\_PREPARE, 309
- TRANSACTION\_STATE\_RECOVER, 309
- TRANSACTION\_STATE\_ROLLBACK, 309
- TransactionState, 308
- URIParam, 309
- activemq::core::ActiveMQConstants::StaticInitializer, 3588
  - ~StaticInitializer, 3588
  - destOptionMap, 3588
  - destOptions, 3588
  - StaticInitializer, 3588
  - uriParams, 3588
  - uriParamsMap, 3588
- activemq::core::ActiveMQConsumer, 310
  - ~ActiveMQConsumer, 312
  - acknowledge, 312
  - ActiveMQConsumer, 312
  - afterMessageIsConsumed, 313
  - beforeMessageIsConsumed, 313
  - clearMessagesInProgress, 313
  - close, 313
  - commit, 313
  - deliverAcks, 314
  - dequeue, 314
  - dispatch, 314
  - doClose, 314
  - getConsumerId, 314
  - getConsumerInfo, 315
  - getLastDeliveredSequenceId, 315
  - getMessageAvailableCount, 315
  - getMessageListener, 315
  - getMessageSelector, 315
  - getRedeliveryPolicy, 316
  - inProgressClearRequired, 316
  - isClosed, 316
  - isSynchronizationRegistered, 316
  - iterate, 316
  - receive, 316, 317
  - receiveNoWait, 317
  - rollback, 317
  - setLastDeliveredSequenceId, 318
  - setMessageListener, 318
  - setRedeliveryPolicy, 318
  - setSynchronizationRegistered, 318
  - start, 318
  - stop, 318
- activemq::core::ActiveMQProducer, 470
  - ~ActiveMQProducer, 472
  - ActiveMQProducer, 471
  - close, 472
  - getDeliveryMode, 472
  - getDisableMessageID, 472
  - getDisableMessageTimeStamp, 472
  - getPriority, 473
  - getProducerId, 473
  - getProducerInfo, 473
  - getSendTimeout, 473
  - getTimeToLive, 473
  - isClosed, 474
  - onProducerAck, 474
  - send, 474, 475
  - setDeliveryMode, 476
  - setDisableMessageID, 476
  - setDisableMessageTimeStamp, 476
  - setPriority, 476
  - setSendTimeout, 477
  - setTimeToLive, 477
- activemq::core::ActiveMQQueueBrowser, 487
  - ~ActiveMQQueueBrowser, 488
  - ActiveMQQueueBrowser, 488
  - Browser, 490
  - close, 488
  - getEnumeration, 488
  - getMessageSelector, 488
  - getQueue, 489
  - hasMoreMessages, 489
  - nextMessage, 489
- activemq::core::ActiveMQSession, 515
  - ~ActiveMQSession, 519
  - acknowledge, 519



- ActiveMQSession, 519
- ActiveMQSessionExecutor, 532
- addConsumer, 519
- addProducer, 519
- clearMessagesInProgress, 519
- close, 520
- commit, 520
- createBrowser, 520
- createBytesMessage, 521
- createConsumer, 521, 522
- createDurableConsumer, 522
- createMapMessage, 523
- createMessage, 523
- createProducer, 523
- createQueue, 524
- createStreamMessage, 524
- createTemporaryQueue, 524
- createTemporaryTopic, 525
- createTextMessage, 525
- createTopic, 525
- deliverAcks, 526
- dispatch, 526
- doStartTransaction, 526
- fire, 526
- getAcknowledgeMode, 526
- getConnection, 527
- getExceptionListener, 527
- getLastDeliveredSequenceId, 527
- getNextConsumerId, 527
- getNextProducerId, 527
- getSessionId, 527
- getSessionInfo, 528
- getTransactionContext, 528
- isAutoAcknowledge, 528
- isClientAcknowledge, 528
- isDupsOkAcknowledge, 528
- isIndividualAcknowledge, 528
- isStarted, 528
- isTransacted, 529
- oneway, 529
- recover, 529
- redispatch, 529
- removeConsumer, 530
- removeProducer, 530
- rollback, 530
- send, 530
- setLastDeliveredSequenceId, 531
- start, 531
- stop, 531
- syncRequest, 531
- unsubscribe, 531
- wakeup, 532
- activemq::core::ActiveMQSessionExecutor, 533
  - ~ActiveMQSessionExecutor, 534
- ActiveMQSessionExecutor, 534
  - clear, 534
  - clearMessagesInProgress, 534
  - close, 534
  - execute, 534
  - executeFirst, 534
  - getUnconsumedMessages, 535
  - hasUnconsumedMessages, 535
  - isEmpty, 535
  - isRunning, 535
  - iterate, 535
  - start, 535
  - stop, 535
  - wakeup, 536
- activemq::core::ActiveMQTransactionContext, 719
  - ~ActiveMQTransactionContext, 720
  - ActiveMQTransactionContext, 720
  - addSynchronization, 720
  - begin, 720
  - commit, 720
  - getTransactionId, 720
  - isInTransaction, 721
  - removeSynchronization, 721
  - rollback, 721
- activemq::core::DispatchData, 1786
  - DispatchData, 1786
  - getConsumerId, 1786
  - getMessage, 1786
- activemq::core::Dispatcher, 1787
  - ~Dispatcher, 1787
  - dispatch, 1787
- activemq::core::MessageDispatchChannel, 2599
  - ~MessageDispatchChannel, 2600
  - clear, 2600
  - close, 2600
  - dequeue, 2600
  - dequeueNoWait, 2601
  - enqueue, 2601
  - enqueueFirst, 2601
  - isClosed, 2601
  - isEmpty, 2601
  - isRunning, 2602
  - lock, 2602
  - MessageDispatchChannel, 2600
  - notify, 2602
  - notifyAll, 2602
  - peek, 2603
  - removeAll, 2603
  - size, 2603
  - start, 2603
  - stop, 2603
  - tryLock, 2603
  - unlock, 2604

- wait, 2604, 2605
- activemq::core::policies, 91
- activemq::core::policies::DefaultPrefetchPolicy, 1673
  - ~DefaultPrefetchPolicy, 1674
  - clone, 1674
  - DEFAULT\_DURABLE\_TOPIC\_PREFETCH, 1676
  - DEFAULT\_QUEUE\_BROWSER\_PREFETCH, 1676
  - DEFAULT\_QUEUE\_PREFETCH, 1676
  - DEFAULT\_TOPIC\_PREFETCH, 1676
  - DefaultPrefetchPolicy, 1674
  - getDurableTopicPrefetch, 1674
  - getMaxPrefetchLimit, 1674
  - getQueueBrowserPrefetch, 1674
  - getQueuePrefetch, 1675
  - getTopicPrefetch, 1675
  - MAX\_PREFETCH\_SIZE, 1676
  - setDurableTopicPrefetch, 1675
  - setQueueBrowserPrefetch, 1675
  - setQueuePrefetch, 1675
  - setTopicPrefetch, 1676
- activemq::core::policies::DefaultRedeliveryPolicy, 1677
  - ~DefaultRedeliveryPolicy, 1678
  - clone, 1678
  - DefaultRedeliveryPolicy, 1678
  - getBackOffMultiplier, 1678
  - getCollisionAvoidancePercent, 1678
  - getInitialRedeliveryDelay, 1678
  - getMaximumRedeliveries, 1678
  - getRedeliveryDelay, 1679
  - isUseCollisionAvoidance, 1679
  - isUseExponentialBackOff, 1679
  - setBackOffMultiplier, 1679
  - setCollisionAvoidancePercent, 1680
  - setInitialRedeliveryDelay, 1680
  - setMaximumRedeliveries, 1680
  - setUseCollisionAvoidance, 1680
  - setUseExponentialBackOff, 1680
- activemq::core::PrefetchPolicy, 2976
  - ~PrefetchPolicy, 2977
  - clone, 2977
  - configure, 2977
  - getDurableTopicPrefetch, 2977
  - getMaxPrefetchLimit, 2978
  - getQueueBrowserPrefetch, 2978
  - getQueuePrefetch, 2978
  - getTopicPrefetch, 2978
  - PrefetchPolicy, 2977
  - setDurableTopicPrefetch, 2978
  - setQueueBrowserPrefetch, 2979
  - setQueuePrefetch, 2979
  - setTopicPrefetch, 2979
- activemq::core::RedeliveryPolicy, 3177
  - ~RedeliveryPolicy, 3178
  - clone, 3178
  - configure, 3178
  - getBackOffMultiplier, 3178
  - getCollisionAvoidancePercent, 3179
  - getInitialRedeliveryDelay, 3179
  - getMaximumRedeliveries, 3179
  - getRedeliveryDelay, 3179
  - isUseCollisionAvoidance, 3180
  - isUseExponentialBackOff, 3180
  - NO\_MAXIMUM\_REDELIVERIES, 3181
  - RedeliveryPolicy, 3178
  - setBackOffMultiplier, 3180
  - setCollisionAvoidancePercent, 3180
  - setInitialRedeliveryDelay, 3180
  - setMaximumRedeliveries, 3180
  - setUseCollisionAvoidance, 3181
  - setUseExponentialBackOff, 3181
- activemq::core::Synchronization, 3715
  - ~Synchronization, 3715
  - afterCommit, 3715
  - afterRollback, 3715
  - beforeEnd, 3715
- activemq::exceptions, 92
- activemq::exceptions::ActiveMQException, 357
  - ~ActiveMQException, 358
  - ActiveMQException, 357, 358
  - clone, 358
  - convertToCMSException, 358
- activemq::exceptions::BrokerException, 867
  - ~BrokerException, 867
  - BrokerException, 867
  - clone, 867
- activemq::io, 93
- activemq::io::LoggingInputStream, 2399
  - ~LoggingInputStream, 2399
  - doReadArrayBounded, 2399
  - doReadByte, 2399
  - LoggingInputStream, 2399
- activemq::io::LoggingOutputStream, 2401
  - ~LoggingOutputStream, 2401
  - doWriteArrayBounded, 2401
  - doWriteByte, 2401
  - LoggingOutputStream, 2401
- activemq::library, 94
- activemq::library::ActiveMQCPP, 320
  - ~ActiveMQCPP, 320
  - ActiveMQCPP, 320
  - initializeLibrary, 320, 321
  - operator=, 321
  - shutdownLibrary, 321
- activemq::state, 95

- activemq::state::CommandVisitor, 1200
  - ~CommandVisitor, 1202
  - processBeginTransaction, 1202
  - processBrokerError, 1202
  - processBrokerInfo, 1202
  - processCommitTransactionOnePhase, 1202
  - processCommitTransactionTwoPhase, 1202
  - processConnectionControl, 1202
  - processConnectionError, 1203
  - processConnectionInfo, 1203
  - processConsumerControl, 1203
  - processConsumerInfo, 1203
  - processControlCommand, 1203
  - processDestinationInfo, 1203
  - processEndTransaction, 1203
  - processFlushCommand, 1203
  - processForgetTransaction, 1204
  - processKeepAliveInfo, 1204
  - processMessage, 1204
  - processMessageAck, 1204
  - processMessageDispatch, 1204
  - processMessageDispatchNotification, 1204
  - processMessagePull, 1204
  - processPrepareTransaction, 1204
  - processProducerAck, 1204
  - processProducerInfo, 1205
  - processRecoverTransactions, 1205
  - processRemoveConnection, 1205
  - processRemoveConsumer, 1205
  - processRemoveDestination, 1205
  - processRemoveInfo, 1205
  - processRemoveProducer, 1205
  - processRemoveSession, 1205
  - processRemoveSubscriptionInfo, 1206
  - processReplayCommand, 1206
  - processResponse, 1206
  - processRollbackTransaction, 1206
  - processSessionInfo, 1206
  - processShutdownInfo, 1206
  - processTransactionInfo, 1206
  - processWireFormat, 1206
- activemq::state::CommandVisitorAdapter, 1208
  - ~CommandVisitorAdapter, 1211
  - processBeginTransaction, 1211
  - processBrokerError, 1211
  - processBrokerInfo, 1211
  - processCommitTransactionOnePhase, 1211
  - processCommitTransactionTwoPhase, 1211
  - processConnectionControl, 1211
  - processConnectionError, 1211
  - processConnectionInfo, 1211
  - processConsumerControl, 1211
  - processConsumerInfo, 1211
  - processControlCommand, 1211
  - processDestinationInfo, 1211
  - processEndTransaction, 1211
  - processFlushCommand, 1211
  - processForgetTransaction, 1211
  - processKeepAliveInfo, 1211
  - processMessage, 1211
  - processMessageAck, 1211
  - processMessageDispatch, 1211
  - processMessageDispatchNotification, 1211
  - processMessagePull, 1211
  - processPrepareTransaction, 1211
  - processProducerAck, 1211
  - processProducerInfo, 1211
  - processRecoverTransactions, 1211
  - processRemoveConnection, 1211
  - processRemoveConsumer, 1211
  - processRemoveDestination, 1211
  - processRemoveInfo, 1211
  - processRemoveProducer, 1212
  - processRemoveSession, 1212
  - processRemoveSubscriptionInfo, 1212
  - processReplayCommand, 1212
  - processResponse, 1212
  - processRollbackTransaction, 1212
  - processSessionInfo, 1212
  - processShutdownInfo, 1212
  - processTransactionInfo, 1212
  - processWireFormat, 1213
- activemq::state::ConnectionState, 1389
  - ~ConnectionState, 1390
  - addSession, 1390
  - addTempDestination, 1390
  - addTransactionState, 1390
  - checkShutdown, 1390
  - ConnectionState, 1390
  - getInfo, 1390
  - getRecoveringPullConsumers, 1390
  - getSessionState, 1390
  - getSessionStates, 1390
  - getTempDestinations, 1390
  - getTransactionState, 1390
  - getTransactionStates, 1390
  - isConnectionInterruptProcessingComplete, 1391
  - removeSession, 1391
  - removeTempDestination, 1391
  - removeTransactionState, 1391
  - reset, 1391
  - setConnectionInterruptProcessingComplete, 1391
  - shutdown, 1391
  - toString, 1391

- activemq::state::ConnectionStateTracker, 1392
  - ~ConnectionStateTracker, 1394
  - connectionInterruptProcessingComplete, 1394
  - ConnectionStateTracker, 1394
  - getMaxCacheSize, 1394
  - isRestoreConsumers, 1394
  - isRestoreProducers, 1394
  - isRestoreSessions, 1394
  - isRestoreTransaction, 1394
  - isTrackMessages, 1394
  - isTrackTransactionProducers, 1394
  - isTrackTransactions, 1394
  - processBeginTransaction, 1394
  - processCommitTransactionOnePhase, 1394
  - processCommitTransactionTwoPhase, 1394
  - processConnectionInfo, 1395
  - processConsumerInfo, 1395
  - processDestinationInfo, 1395
  - processEndTransaction, 1395
  - processMessage, 1395
  - processMessageAck, 1395
  - processPrepareTransaction, 1395
  - processProducerInfo, 1396
  - processRemoveConnection, 1396
  - processRemoveConsumer, 1396
  - processRemoveDestination, 1396
  - processRemoveProducer, 1396
  - processRemoveSession, 1396
  - processRollbackTransaction, 1396
  - processSessionInfo, 1397
  - RemoveTransactionAction, 1398
  - restore, 1397
  - setMaxCacheSize, 1398
  - setRestoreConsumers, 1398
  - setRestoreProducers, 1398
  - setRestoreSessions, 1398
  - setRestoreTransaction, 1398
  - setTrackMessages, 1398
  - setTrackTransactionProducers, 1398
  - setTrackTransactions, 1398
  - track, 1398
  - trackBack, 1398
  - transportInterrupted, 1398
- activemq::state::ConsumerState, 1492
  - ~ConsumerState, 1492
  - ConsumerState, 1492
  - getInfo, 1492
  - toString, 1492
- activemq::state::ProducerState, 3125
  - ~ProducerState, 3125
  - getInfo, 3125
  - getTransactionState, 3125
  - ProducerState, 3125
  - setTransactionState, 3125
  - toString, 3125
- activemq::state::SessionState, 3437
  - ~SessionState, 3438
  - addConsumer, 3438
  - addProducer, 3438
  - checkShutdown, 3438
  - getConsumerState, 3438
  - getConsumerStates, 3438
  - getInfo, 3438
  - getProducerState, 3438
  - getProducerStates, 3438
  - removeConsumer, 3438
  - removeProducer, 3438
  - SessionState, 3438
  - shutdown, 3438
  - toString, 3438
- activemq::state::Tracked, 3818
  - ~Tracked, 3818
  - isWaitingForResponse, 3818
  - onResponse, 3818
  - Tracked, 3818
- activemq::state::TransactionState, 3875
  - ~TransactionState, 3876
  - addCommand, 3876
  - addProducerState, 3876
  - checkShutdown, 3876
  - getCommands, 3876
  - getId, 3876
  - getPreparedResult, 3876
  - getProducerStates, 3876
  - isPrepared, 3876
  - setPrepared, 3876
  - setPreparedResult, 3876
  - shutdown, 3876
  - toString, 3876
  - TransactionState, 3876
- activemq::threads, 96
- activemq::threads::CompositeTask, 1221
  - ~CompositeTask, 1221
  - isPending, 1221
- activemq::threads::CompositeTaskRunner, 1223
  - ~CompositeTaskRunner, 1224
  - addTask, 1224
  - CompositeTaskRunner, 1224
  - iterate, 1224
  - removeTask, 1224
  - run, 1224
  - shutdown, 1224
  - wakeup, 1225
- activemq::threads::DedicatedTaskRunner, 1671
  - ~DedicatedTaskRunner, 1671
  - DedicatedTaskRunner, 1671

- run, 1671
- shutdown, 1671, 1672
- wakeup, 1672
- activemq::threads::Task, 3734
  - ~Task, 3734
  - iterate, 3734
- activemq::threads::TaskRunner, 3736
  - ~TaskRunner, 3736
  - shutdown, 3736
  - wakeup, 3736
- activemq::transport, 97
- activemq::transport::AbstractTransportFactory, 201
  - ~AbstractTransportFactory, 201
  - createWireFormat, 201
- activemq::transport::CompositeTransport, 1226
  - ~CompositeTransport, 1226
  - addURI, 1226
  - removeURI, 1226
- activemq::transport::correlator, 98
- activemq::transport::correlator::FutureResponse, 1969
  - ~FutureResponse, 1969
  - FutureResponse, 1969
  - getResponse, 1969, 1970
  - setResponse, 1970
- activemq::transport::correlator::ResponseCorrelator, 3291
  - ~ResponseCorrelator, 3292
  - close, 3292
  - onCommand, 3292
  - oneway, 3292
  - onTransportException, 3293
  - request, 3293
  - ResponseCorrelator, 3292
  - start, 3294
- activemq::transport::DefaultTransportListener, 1704
  - ~DefaultTransportListener, 1704
  - onCommand, 1704
  - onException, 1704
  - transportInterrupted, 1705
  - transportResumed, 1705
- activemq::transport::failover, 99
- activemq::transport::failover::BackupTransport, 752
  - ~BackupTransport, 752
  - BackupTransport, 752
  - getTransport, 752
  - getUri, 753
  - isClosed, 753
  - onException, 753
  - setClosed, 753
  - setTransport, 753
  - setUri, 753
- activemq::transport::failover::BackupTransportPool, 755
  - ~BackupTransportPool, 756
  - BackupTransport, 757
  - BackupTransportPool, 756
  - getBackup, 756
  - getBackupPoolSize, 756
  - isEnabled, 756
  - isPending, 756
  - iterate, 757
  - setBackupPoolSize, 757
  - setEnabled, 757
- activemq::transport::failover::CloseTransportsTask, 1151
  - ~CloseTransportsTask, 1151
  - add, 1151
  - CloseTransportsTask, 1151
  - isPending, 1151
  - iterate, 1151
- activemq::transport::failover::FailoverTransport, 1873
  - ~FailoverTransport, 1875
  - add, 1875
  - addURI, 1876
  - close, 1876
  - FailoverTransport, 1875
  - FailoverTransportListener, 1884
  - getBackOffMultiplier, 1876
  - getBackupPoolSize, 1877
  - getInitialReconnectDelay, 1877
  - getMaxCacheSize, 1877
  - getMaxReconnectAttempts, 1877
  - getMaxReconnectDelay, 1877
  - getReconnectDelay, 1877
  - getRemoteAddress, 1877
  - getStartupMaxReconnectAttempts, 1877
  - getTimeout, 1877
  - getTransportListener, 1877
  - handleTransportFailure, 1877
  - isBackup, 1878
  - isClosed, 1878
  - isConnected, 1878
  - isFaultTolerant, 1878
  - isInitialized, 1878
  - isPending, 1879
  - isRandomize, 1879
  - isTrackMessages, 1879
  - isTrackTransactionProducers, 1879
  - isUseExponentialBackOff, 1879
  - iterate, 1879
  - narrow, 1879
  - oneway, 1880
  - reconnect, 1880

- removeURI, 1880
- request, 1881
- restoreTransport, 1882
- setBackOffMultiplier, 1882
- setBackup, 1882
- setBackupPoolSize, 1882
- setConnectionInterruptProcessingComplete, 1882
- setInitialized, 1882
- setInitialReconnectDelay, 1882
- setMaxCacheSize, 1883
- setMaxReconnectAttempts, 1883
- setMaxReconnectDelay, 1883
- setRandomize, 1883
- setReconnectDelay, 1883
- setStartupMaxReconnectAttempts, 1883
- setTimeout, 1883
- setTrackMessages, 1883
- setTrackTransactionProducers, 1883
- setTransportListener, 1883
- setUseExponentialBackOff, 1883
- setWireFormat, 1884
- start, 1884
- stop, 1884
- activemq::transport::failover::FailoverTransportFactory, 1885
  - ~FailoverTransportFactory, 1886
  - create, 1886
  - createComposite, 1886
  - doCreateComposite, 1886
- activemq::transport::failover::FailoverTransportListener, 1888
  - ~FailoverTransportListener, 1889
  - FailoverTransportListener, 1889
  - onCommand, 1889
  - onException, 1889
  - transportInterrupted, 1889
  - transportResumed, 1889
- activemq::transport::failover::URIPool, 3942
  - ~URIPool, 3942
  - addURI, 3943
  - addURIs, 3943
  - getURI, 3943
  - isRandomize, 3943
  - removeURI, 3943
  - setRandomize, 3944
  - URIPool, 3942
- activemq::transport::inactivity, 100
- activemq::transport::inactivity::InactivityMonitor, 2004
  - ~InactivityMonitor, 2005
  - AsyncSignalReadErrorTask, 2007
  - AsyncWriteTask, 2007
  - close, 2005
  - getInitialDelayTime, 2005
  - getReadCheckTime, 2005
  - getWriteCheckTime, 2005
  - InactivityMonitor, 2005
  - isKeepAliveResponseRequired, 2005
  - onCommand, 2005
  - oneway, 2006
  - onException, 2006
  - ReadChecker, 2007
  - setInitialDelayTime, 2006
  - setKeepAliveResponseRequired, 2007
  - setReadCheckTime, 2007
  - setWriteCheckTime, 2007
  - WriteChecker, 2007
- activemq::transport::inactivity::ReadChecker, 3162
  - ~ReadChecker, 3162
  - ReadChecker, 3162
  - run, 3162
- activemq::transport::inactivity::WriteChecker, 4020
  - ~WriteChecker, 4020
  - run, 4020
  - WriteChecker, 4020
- activemq::transport::IOTransport, 2145
  - ~IOTransport, 2147
  - close, 2147
  - getRemoteAddress, 2147
  - getTransportListener, 2147
  - IOTransport, 2146
  - isClosed, 2147
  - isConnected, 2148
  - isFaultTolerant, 2148
  - narrow, 2148
  - oneway, 2148
  - reconnect, 2148
  - request, 2149
  - run, 2149
  - setInputStream, 2150
  - setOutputStream, 2150
  - setTransportListener, 2150
  - setWireFormat, 2150
  - start, 2150
  - stop, 2151
- activemq::transport::logging, 101
- activemq::transport::logging::LoggingTransport, 2403
  - ~LoggingTransport, 2403
  - LoggingTransport, 2403
  - onCommand, 2404
  - oneway, 2404
  - request, 2404
- activemq::transport::mock, 102

- activemq::transport::mock::InternalCommandListener, start, 2777
  - 2122
  - ~InternalCommandListener, 2122
  - InternalCommandListener, 2122
  - onCommand, 2122
  - run, 2123
  - setResponseBuilder, 2123
  - setTransport, 2123
- activemq::transport::mock::MockTransport, 2768
  - ~MockTransport, 2770
  - close, 2770
  - fireCommand, 2770
  - fireException, 2771
  - getInstance, 2771
  - getNumReceivedMessageBeforeFail, 2771
  - getNumReceivedMessages, 2771
  - getNumSentKeepAlives, 2771
  - getNumSentKeepAlivesBeforeFail, 2771
  - getNumSentMessageBeforeFail, 2771
  - getNumSentMessages, 2771
  - getRemoteAddress, 2771
  - getTransportListener, 2772
  - getWireFormat, 2772
  - isClosed, 2772
  - isConnected, 2772
  - isFailOnClose, 2772
  - isFailOnKeepAliveSends, 2773
  - isFailOnReceiveMessage, 2773
  - isFailOnSendMessage, 2773
  - isFailOnStart, 2773
  - isFailOnStop, 2773
  - isFaultTolerant, 2773
  - MockTransport, 2770
  - narrow, 2773
  - oneway, 2773
  - reconnect, 2774
  - request, 2774
  - setFailOnClose, 2775
  - setFailOnKeepAliveSends, 2776
  - setFailOnReceiveMessage, 2776
  - setFailOnSendMessage, 2776
  - setFailOnStart, 2776
  - setFailOnStop, 2776
  - setNumReceivedMessageBeforeFail, 2776
  - setNumReceivedMessages, 2776
  - setNumSentKeepAlives, 2776
  - setNumSentKeepAlivesBeforeFail, 2776
  - setNumSentMessageBeforeFail, 2776
  - setNumSentMessages, 2776
  - setOutgoingListener, 2776
  - setResponseBuilder, 2776
  - setTransportListener, 2777
  - setWireFormat, 2777
- activemq::transport::mock::MockTransportFactory, 2779
  - ~MockTransportFactory, 2780
  - create, 2780
  - createComposite, 2780
  - doCreateComposite, 2780
- activemq::transport::mock::ResponseBuilder, 3289
  - ~ResponseBuilder, 3289
  - buildIncomingCommands, 3289
  - buildResponse, 3290
- activemq::transport::tcp, 103
- activemq::transport::tcp::SslTransport, 3574
  - ~SslTransport, 3575
  - configureSocket, 3575
  - createSocket, 3575
  - SslTransport, 3574
- activemq::transport::tcp::SslTransportFactory, 3576
  - ~SslTransportFactory, 3576
  - doCreateComposite, 3576
- activemq::transport::tcp::TcpTransport, 3752
  - ~TcpTransport, 3753
  - close, 3753
  - configureSocket, 3753
  - connect, 3753
  - createSocket, 3754
  - isClosed, 3754
  - isConnected, 3754
  - isFaultTolerant, 3754
  - TcpTransport, 3753
- activemq::transport::tcp::TcpTransportFactory, 3756
  - ~TcpTransportFactory, 3757
  - create, 3757
  - createComposite, 3757
  - doCreateComposite, 3757
- activemq::transport::Transport, 3883
  - ~Transport, 3884
  - getRemoteAddress, 3884
  - getTransportListener, 3884
  - isClosed, 3884
  - isConnected, 3885
  - isFaultTolerant, 3885
  - narrow, 3885
  - oneway, 3886
  - reconnect, 3886
  - request, 3886, 3887
  - setTransportListener, 3887
  - setWireFormat, 3888
  - start, 3888
  - stop, 3888

- activemq::transport::TransportFactory, 3889
  - ~TransportFactory, 3889
  - create, 3889
  - createComposite, 3890
- activemq::transport::TransportFilter, 3891
  - ~TransportFilter, 3893
  - close, 3893
  - fire, 3893
  - getRemoteAddress, 3894
  - getTransportListener, 3894
  - isClosed, 3894
  - isConnected, 3894
  - isFaultTolerant, 3894
  - listener, 3898
  - narrow, 3895
  - next, 3898
  - onCommand, 3895
  - oneway, 3895
  - onException, 3896
  - reconnect, 3896
  - request, 3896, 3897
  - setTransportListener, 3897
  - setWireFormat, 3897
  - start, 3897
  - stop, 3898
  - TransportFilter, 3893
  - transportInterrupted, 3898
  - transportResumed, 3898
- activemq::transport::TransportListener, 3900
  - ~TransportListener, 3900
  - onCommand, 3900
  - onException, 3901
  - transportInterrupted, 3901
  - transportResumed, 3901
- activemq::transport::TransportRegistry, 3902
  - ~TransportRegistry, 3903
  - findFactory, 3903
  - getInstance, 3903
  - getTransportNames, 3903
  - registerFactory, 3903
  - unregisterFactory, 3904
- activemq::util, 104
- activemq::util::ActiveMQProperties, 478
  - ~ActiveMQProperties, 479
  - ActiveMQProperties, 479
  - clear, 479
  - clone, 479
  - copy, 479
  - getProperties, 479, 480
  - getProperty, 480
  - hasProperty, 480
  - isEmpty, 480
  - remove, 481
  - setProperty, 481
  - setProperty, 481
  - toArray, 481
  - toString, 481
- activemq::util::CMSExceptionSupport, 1163
  - ~CMSExceptionSupport, 1163
  - create, 1163
  - createMessageEOFException, 1163
  - createMessageFormatException, 1163
- activemq::util::CompositeData, 1219
  - ~CompositeData, 1220
  - CompositeData, 1220
  - getComponents, 1220
  - getFragment, 1220
  - getHost, 1220
  - getParameters, 1220
  - getPath, 1220
  - getScheme, 1220
  - setComponents, 1220
  - setFragment, 1220
  - setHost, 1220
  - setParameters, 1220
  - setPath, 1220
  - setScheme, 1220
  - toURI, 1220
- activemq::util::IdGenerator, 1989
  - ~IdGenerator, 1989
  - compare, 1989
  - generateId, 1990
  - getHostname, 1990
  - getSeedFromId, 1990
  - getSequenceFromId, 1990
  - IdGenerator, 1989
- activemq::util::LongSequenceGenerator, 2455
  - ~LongSequenceGenerator, 2455
  - getLastSequenceId, 2455
  - getNextSequenceId, 2455
  - LongSequenceGenerator, 2455
- activemq::util::MarshallingSupport, 2493
  - ~MarshallingSupport, 2494
  - asciiToModifiedUtf8, 2494
  - MarshallingSupport, 2494
  - modifiedUtf8ToAscii, 2494
  - readString16, 2494
  - readString32, 2495
  - writeString, 2495
  - writeString16, 2496
  - writeString32, 2496
- activemq::util::MemoryUsage, 2512
  - ~MemoryUsage, 2513
  - decreaseUsage, 2513
  - enqueueUsage, 2513
  - getLimit, 2513
  - getUsage, 2513
  - increaseUsage, 2514



- isFull, 2514
- MemoryUsage, 2513
- setLimit, 2514
- setUsage, 2514
- waitForSpace, 2514
- activemq::util::PrimitiveList, 2980
  - ~PrimitiveList, 2982
  - getBool, 2983
  - getByte, 2983
  - getByteArray, 2983
  - getChar, 2984
  - getDouble, 2984
  - getFloat, 2985
  - getInt, 2985
  - getLong, 2985
  - getShort, 2986
  - getString, 2986
  - PrimitiveList, 2982
  - setBool, 2986
  - setByte, 2987
  - setByteArray, 2987
  - setChar, 2987
  - setDouble, 2988
  - setFloat, 2988
  - setInt, 2988
  - setLong, 2989
  - setShort, 2989
  - setString, 2989
  - toString, 2990
- activemq::util::PrimitiveMap, 2991
  - ~PrimitiveMap, 2993
  - getBool, 2994
  - getByte, 2994
  - getByteArray, 2994
  - getChar, 2995
  - getDouble, 2995
  - getFloat, 2995
  - getInt, 2996
  - getLong, 2996
  - getShort, 2997
  - getString, 2997
  - PrimitiveMap, 2993
  - setBool, 2997
  - setByte, 2998
  - setByteArray, 2998
  - setChar, 2998
  - setDouble, 2998
  - setFloat, 2999
  - setInt, 2999
  - setLong, 2999
  - setShort, 2999
  - setString, 2999
  - toString, 3000
- activemq::util::PrimitiveValueConverter, 3010
  - ~PrimitiveValueConverter, 3010
  - convert, 3010
  - PrimitiveValueConverter, 3010
- activemq::util::PrimitiveValueNode, 3012
  - ~PrimitiveValueNode, 3018
  - BIG\_STRING\_TYPE, 3016
  - BOOLEAN\_TYPE, 3015
  - BYTE\_ARRAY\_TYPE, 3016
  - BYTE\_TYPE, 3016
  - CHAR\_TYPE, 3016
  - clear, 3018
  - DOUBLE\_TYPE, 3016
  - FLOAT\_TYPE, 3016
  - getBool, 3018
  - getByte, 3019
  - getByteArray, 3019
  - getChar, 3019
  - getDouble, 3019
  - getFloat, 3020
  - getInt, 3020
  - getList, 3020
  - getLong, 3020
  - getMap, 3021
  - getShort, 3021
  - getString, 3021
  - getType, 3021
  - getValue, 3022
  - INTEGER\_TYPE, 3016
  - LIST\_TYPE, 3016
  - LONG\_TYPE, 3016
  - MAP\_TYPE, 3016
  - NULL\_TYPE, 3015
  - operator=, 3022
  - operator==, 3022
  - PrimitiveType, 3015
  - PrimitiveValueNode, 3016–3018
  - setBool, 3022
  - setByte, 3022
  - setByteArray, 3022
  - setChar, 3023
  - setDouble, 3023
  - setFloat, 3023
  - setInt, 3023
  - setList, 3023
  - setLong, 3024
  - setMap, 3024
  - setShort, 3024
  - setString, 3024
  - setValue, 3024
  - SHORT\_TYPE, 3016
  - STRING\_TYPE, 3016
  - toString, 3024
- activemq::util::PrimitiveValueNode::PrimitiveValue, 3008

- boolValue, 3009
- byteArrayValue, 3009
- byteValue, 3009
- charValue, 3009
- doubleValue, 3009
- floatValue, 3009
- intValue, 3009
- listValue, 3009
- longValue, 3009
- mapValue, 3009
- shortValue, 3009
- stringValue, 3009
- activemq::util::URISupport, 3945
  - createQueryString, 3945
  - parseComposite, 3945
  - parseQuery, 3946
  - parseURL, 3946
- activemq::util::Usage, 3964
  - ~Usage, 3964
  - decreaseUsage, 3964
  - enqueueUsage, 3964
  - increaseUsage, 3965
  - isFull, 3965
  - waitForSpace, 3965
- activemq::wireformat, 105
- activemq::wireformat::MarshalAware, 2484
  - ~MarshalAware, 2484
  - afterMarshal, 2484
  - afterUnmarshal, 2485
  - beforeMarshal, 2485
  - beforeUnmarshal, 2485
  - getMarshaledForm, 2485
  - isMarshalAware, 2485
  - setMarshaledForm, 2486
- activemq::wireformat::openwire, 106
- activemq::wireformat::openwire::marshal, 107
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 807
  - ~BaseDataStreamMarshaller, 813
  - looseMarshal, 813
  - looseMarshalBrokerError, 814
  - looseMarshalCachedObject, 814
  - looseMarshalLong, 814
  - looseMarshalNestedObject, 815
  - looseMarshalObjectArray, 815
  - looseMarshalString, 815
  - looseUnmarshal, 816
  - looseUnmarshalBrokerError, 816
  - looseUnmarshalByteArray, 816
  - looseUnmarshalCachedObject, 817
  - looseUnmarshalConstByteArray, 817
  - looseUnmarshalLong, 818
  - looseUnmarshalNestedObject, 818
  - looseUnmarshalString, 818
- readAsciiString, 819
- tightMarshal1, 819
- tightMarshal2, 819
- tightMarshalBrokerError1, 820
- tightMarshalBrokerError2, 820
- tightMarshalCachedObject1, 820
- tightMarshalCachedObject2, 821
- tightMarshalLong1, 821
- tightMarshalLong2, 822
- tightMarshalNestedObject1, 822
- tightMarshalNestedObject2, 822
- tightMarshalObjectArray1, 823
- tightMarshalObjectArray2, 823
- tightMarshalString1, 824
- tightMarshalString2, 824
- tightUnmarshal, 824
- tightUnmarshalBrokerError, 825
- tightUnmarshalByteArray, 825
- tightUnmarshalCachedObject, 826
- tightUnmarshalConstByteArray, 826
- tightUnmarshalLong, 826
- tightUnmarshalNestedObject, 827
- tightUnmarshalString, 827
- toHexFromBytes, 828
- toString, 828
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1610
  - ~DataStreamMarshaller, 1611
  - createObject, 1611
  - getDataStructureType, 1617
  - looseMarshal, 1623
  - looseUnmarshal, 1631
  - tightMarshal1, 1638
  - tightMarshal2, 1645
  - tightUnmarshal, 1652
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3001
  - ~PrimitiveTypesMarshaller, 3003
  - marshal, 3003
  - marshalList, 3003
  - marshalMap, 3004
  - marshalPrimitive, 3004
  - marshalPrimitiveList, 3004
  - marshalPrimitiveMap, 3004
  - PrimitiveTypesMarshaller, 3003
  - unmarshal, 3005
  - unmarshalList, 3005
  - unmarshalMap, 3006
  - unmarshalPrimitive, 3006
  - unmarshalPrimitiveList, 3006
  - unmarshalPrimitiveMap, 3007
- activemq::wireformat::openwire::marshal::v1, 108

- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 213
  - ~ActiveMQBlobMessageMarshaller, 214
  - ActiveMQBlobMessageMarshaller, 214
  - createObject, 214
  - getDataStructureType, 214
  - looseMarshal, 214
  - looseUnmarshal, 214
  - tightMarshal1, 215
  - tightMarshal2, 215
  - tightUnmarshal, 216
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 253
  - ~ActiveMQBytesMessageMarshaller, 254
  - ActiveMQBytesMessageMarshaller, 254
  - createObject, 254
  - getDataStructureType, 254
  - looseMarshal, 254
  - looseUnmarshal, 254
  - tightMarshal1, 255
  - tightMarshal2, 255
  - tightUnmarshal, 256
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 337
  - ~ActiveMQDestinationMarshaller, 338
  - ActiveMQDestinationMarshaller, 338
  - looseMarshal, 338
  - looseUnmarshal, 338
  - tightMarshal1, 339
  - tightMarshal2, 339
  - tightUnmarshal, 340
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 378
  - ~ActiveMQMapMessageMarshaller, 379
  - ActiveMQMapMessageMarshaller, 379
  - createObject, 379
  - getDataStructureType, 379
  - looseMarshal, 379
  - looseUnmarshal, 379
  - tightMarshal1, 380
  - tightMarshal2, 380
  - tightUnmarshal, 381
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 405
  - ~ActiveMQMessageMarshaller, 406
  - ActiveMQMessageMarshaller, 406
  - createObject, 406
  - getDataStructureType, 406
  - looseMarshal, 406
  - looseUnmarshal, 406
  - tightMarshal1, 407
  - tightMarshal2, 407
  - tightUnmarshal, 408
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 450
  - ~ActiveMQObjectMessageMarshaller, 451
  - ActiveMQObjectMessageMarshaller, 451
  - createObject, 451
  - getDataStructureType, 451
  - looseMarshal, 451
  - looseUnmarshal, 451
  - tightMarshal1, 452
  - tightMarshal2, 452
  - tightUnmarshal, 453
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 495
  - ~ActiveMQQueueMarshaller, 496
  - ActiveMQQueueMarshaller, 496
  - createObject, 496
  - getDataStructureType, 496
  - looseMarshal, 496
  - looseUnmarshal, 496
  - tightMarshal1, 497
  - tightMarshal2, 497
  - tightUnmarshal, 498
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 556
  - ~ActiveMQStreamMessageMarshaller, 557
  - ActiveMQStreamMessageMarshaller, 557
  - createObject, 557
  - getDataStructureType, 557
  - looseMarshal, 557
  - looseUnmarshal, 557
  - tightMarshal1, 558
  - tightMarshal2, 558
  - tightUnmarshal, 559
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 584
  - ~ActiveMQTempDestinationMarshaller, 585
  - ActiveMQTempDestinationMarshaller, 585
  - looseMarshal, 585
  - looseUnmarshal, 585
  - tightMarshal1, 586
  - tightMarshal2, 586
  - tightUnmarshal, 587
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 613
  - ~ActiveMQTempQueueMarshaller, 614
  - ActiveMQTempQueueMarshaller, 614
  - createObject, 614
  - getDataStructureType, 614
  - looseMarshal, 614
  - looseUnmarshal, 614
  - tightMarshal1, 615
  - tightMarshal2, 615
  - tightUnmarshal, 616

- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 646
  - ~ActiveMQTempTopicMarshaller, 647
  - ActiveMQTempTopicMarshaller, 647
  - createObject, 647
  - getDataStructureType, 647
  - looseMarshal, 647
  - looseUnmarshal, 647
  - tightMarshal1, 648
  - tightMarshal2, 648
  - tightUnmarshal, 649
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 675
  - ~ActiveMQTextMessageMarshaller, 676
  - ActiveMQTextMessageMarshaller, 676
  - createObject, 676
  - getDataStructureType, 676
  - looseMarshal, 676
  - looseUnmarshal, 676
  - tightMarshal1, 677
  - tightMarshal2, 677
  - tightUnmarshal, 678
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 703
  - ~ActiveMQTopicMarshaller, 704
  - ActiveMQTopicMarshaller, 704
  - createObject, 704
  - getDataStructureType, 704
  - looseMarshal, 704
  - looseUnmarshal, 704
  - tightMarshal1, 705
  - tightMarshal2, 705
  - tightUnmarshal, 706
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 779
  - ~BaseCommandMarshaller, 780
  - BaseCommandMarshaller, 780
  - looseMarshal, 780
  - looseUnmarshal, 781
  - tightMarshal1, 782
  - tightMarshal2, 783
  - tightUnmarshal, 784
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 880
  - ~BrokerIdMarshaller, 881
  - BrokerIdMarshaller, 881
  - createObject, 881
  - getDataStructureType, 881
  - looseMarshal, 881
  - looseUnmarshal, 881
  - tightMarshal1, 882
  - tightMarshal2, 882
  - tightUnmarshal, 883
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 912
  - ~BrokerInfoMarshaller, 913
  - BrokerInfoMarshaller, 913
  - createObject, 913
  - getDataStructureType, 913
  - looseMarshal, 913
  - looseUnmarshal, 913
  - tightMarshal1, 914
  - tightMarshal2, 914
  - tightUnmarshal, 915
- activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1280
  - ~ConnectionControlMarshaller, 1281
  - ConnectionControlMarshaller, 1281
  - createObject, 1281
  - getDataStructureType, 1281
  - looseMarshal, 1281
  - looseUnmarshal, 1281
  - tightMarshal1, 1282
  - tightMarshal2, 1282
  - tightUnmarshal, 1283
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1312
  - ~ConnectionErrorMarshaller, 1313
  - ConnectionErrorMarshaller, 1313
  - createObject, 1313
  - getDataStructureType, 1313
  - looseMarshal, 1313
  - looseUnmarshal, 1313
  - tightMarshal1, 1314
  - tightMarshal2, 1314
  - tightUnmarshal, 1315
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1343
  - ~ConnectionIdMarshaller, 1344
  - ConnectionIdMarshaller, 1344
  - createObject, 1344
  - getDataStructureType, 1344
  - looseMarshal, 1344
  - looseUnmarshal, 1344
  - tightMarshal1, 1345
  - tightMarshal2, 1345
  - tightUnmarshal, 1346
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1373
  - ~ConnectionInfoMarshaller, 1374
  - ConnectionInfoMarshaller, 1374
  - createObject, 1374
  - getDataStructureType, 1374
  - looseMarshal, 1374
  - looseUnmarshal, 1374
  - tightMarshal1, 1375
  - tightMarshal2, 1375

- tightUnmarshal, 1376
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1418
  - ~ConsumerControlMarshaller, 1419
  - ConsumerControlMarshaller, 1419
  - createObject, 1419
  - getDataStructureType, 1419
  - looseMarshal, 1419
  - looseUnmarshal, 1419
  - tightMarshal1, 1420
  - tightMarshal2, 1420
  - tightUnmarshal, 1421
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1447
  - ~ConsumerIdMarshaller, 1448
  - ConsumerIdMarshaller, 1448
  - createObject, 1448
  - getDataStructureType, 1448
  - looseMarshal, 1448
  - looseUnmarshal, 1448
  - tightMarshal1, 1449
  - tightMarshal2, 1449
  - tightUnmarshal, 1450
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1480
  - ~ConsumerInfoMarshaller, 1481
  - ConsumerInfoMarshaller, 1481
  - createObject, 1481
  - getDataStructureType, 1481
  - looseMarshal, 1481
  - looseUnmarshal, 1481
  - tightMarshal1, 1482
  - tightMarshal2, 1482
  - tightUnmarshal, 1483
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1509
  - ~ControlCommandMarshaller, 1510
  - ControlCommandMarshaller, 1510
  - createObject, 1510
  - getDataStructureType, 1510
  - looseMarshal, 1510
  - looseUnmarshal, 1510
  - tightMarshal1, 1511
  - tightMarshal2, 1511
  - tightUnmarshal, 1512
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1543
  - ~DataArrayResponseMarshaller, 1544
  - createObject, 1544
  - DataArrayResponseMarshaller, 1544
  - getDataStructureType, 1544
  - looseMarshal, 1544
  - looseUnmarshal, 1545
  - tightMarshal1, 1545
- tightMarshal2, 1545
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1606
  - ~DataResponseMarshaller, 1607
  - createObject, 1607
  - DataResponseMarshaller, 1607
  - getDataStructureType, 1607
  - looseMarshal, 1607
  - looseUnmarshal, 1608
  - tightMarshal1, 1608
  - tightMarshal2, 1608
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1744
  - ~DestinationInfoMarshaller, 1745
  - createObject, 1745
  - DestinationInfoMarshaller, 1745
  - getDataStructureType, 1745
  - looseMarshal, 1745
  - looseUnmarshal, 1745
  - tightMarshal1, 1746
  - tightMarshal2, 1746
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1778
  - ~DiscoveryEventMarshaller, 1779
  - createObject, 1779
  - DiscoveryEventMarshaller, 1779
  - getDataStructureType, 1779
  - looseMarshal, 1779
  - looseUnmarshal, 1779
  - tightMarshal1, 1780
  - tightMarshal2, 1780
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1862
  - ~ExceptionResponseMarshaller, 1863
  - createObject, 1863
  - ExceptionResponseMarshaller, 1863
  - getDataStructureType, 1863
  - looseMarshal, 1863
  - looseUnmarshal, 1864
  - tightMarshal1, 1864
  - tightMarshal2, 1864
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1956
  - ~FlushCommandMarshaller, 1957
  - createObject, 1957
  - FlushCommandMarshaller, 1957
  - getDataStructureType, 1957
  - looseMarshal, 1957
  - looseUnmarshal, 1957

- tightMarshal1, 1958
- tightMarshal2, 1958
- tightUnmarshal, 1959
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2110
  - ~IntegerResponseMarshaller, 2111
  - createObject, 2111
  - getDataStructureType, 2111
  - IntegerResponseMarshaller, 2111
  - looseMarshal, 2111
  - looseUnmarshal, 2112
  - tightMarshal1, 2112
  - tightMarshal2, 2112
  - tightUnmarshal, 2113
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2179
  - ~JournalQueueAckMarshaller, 2180
  - createObject, 2180
  - getDataStructureType, 2180
  - JournalQueueAckMarshaller, 2180
  - looseMarshal, 2180
  - looseUnmarshal, 2180
  - tightMarshal1, 2181
  - tightMarshal2, 2181
  - tightUnmarshal, 2182
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2208
  - ~JournalTopicAckMarshaller, 2209
  - createObject, 2209
  - getDataStructureType, 2209
  - JournalTopicAckMarshaller, 2209
  - looseMarshal, 2209
  - looseUnmarshal, 2209
  - tightMarshal1, 2210
  - tightMarshal2, 2210
  - tightUnmarshal, 2211
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2231
  - ~JournalTraceMarshaller, 2232
  - createObject, 2232
  - getDataStructureType, 2232
  - JournalTraceMarshaller, 2232
  - looseMarshal, 2232
  - looseUnmarshal, 2232
  - tightMarshal1, 2233
  - tightMarshal2, 2233
  - tightUnmarshal, 2234
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2262
  - ~JournalTransactionMarshaller, 2263
  - createObject, 2263
  - getDataStructureType, 2263
  - JournalTransactionMarshaller, 2263
  - looseMarshal, 2263
  - looseUnmarshal, 2263
  - tightMarshal1, 2264
  - tightMarshal2, 2264
  - tightUnmarshal, 2265
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2289
  - ~KeepAliveInfoMarshaller, 2290
  - createObject, 2290
  - getDataStructureType, 2290
  - KeepAliveInfoMarshaller, 2290
  - looseMarshal, 2290
  - looseUnmarshal, 2290
  - tightMarshal1, 2291
  - tightMarshal2, 2291
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2324
  - ~LastPartialCommandMarshaller, 2325
  - createObject, 2325
  - getDataStructureType, 2325
  - LastPartialCommandMarshaller, 2325
  - looseMarshal, 2325
  - looseUnmarshal, 2326
  - tightMarshal1, 2326
  - tightMarshal2, 2326
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2371
  - ~LocalTransactionIdMarshaller, 2372
  - createObject, 2372
  - getDataStructureType, 2372
  - LocalTransactionIdMarshaller, 2372
  - looseMarshal, 2372
  - looseUnmarshal, 2372
  - tightMarshal1, 2373
  - tightMarshal2, 2373
- activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2491
  - ~MarshallerFactory, 2491
  - configure, 2491
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2581
  - ~MessageAckMarshaller, 2582
  - createObject, 2582
  - getDataStructureType, 2582
  - looseMarshal, 2582
  - looseUnmarshal, 2582
  - MessageAckMarshaller, 2582
  - tightMarshal1, 2583
  - tightMarshal2, 2583
  - tightUnmarshal, 2584
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2622

- ~MessageDispatchMarshaller, 2623
  - createObject, 2623
  - getDataStructureType, 2623
  - looseMarshal, 2623
  - looseUnmarshal, 2623
  - MessageDispatchMarshaller, 2623
  - tightMarshal1, 2624
  - tightMarshal2, 2624
  - tightUnmarshal, 2625
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2651
  - ~MessageDispatchNotificationMarshaller, 2652
  - createObject, 2652
  - getDataStructureType, 2652
  - looseMarshal, 2652
  - looseUnmarshal, 2652
  - MessageDispatchNotificationMarshaller, 2652
  - tightMarshal1, 2653
  - tightMarshal2, 2653
  - tightUnmarshal, 2654
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2688
  - ~MessageIdMarshaller, 2689
  - createObject, 2689
  - getDataStructureType, 2689
  - looseMarshal, 2689
  - looseUnmarshal, 2689
  - MessageIdMarshaller, 2689
  - tightMarshal1, 2690
  - tightMarshal2, 2690
  - tightUnmarshal, 2691
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2713
  - ~MessageMarshaller, 2714
  - looseMarshal, 2714
  - looseUnmarshal, 2714
  - MessageMarshaller, 2714
  - tightMarshal1, 2715
  - tightMarshal2, 2715
  - tightUnmarshal, 2716
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2760
  - ~MessagePullMarshaller, 2761
  - createObject, 2761
  - getDataStructureType, 2761
  - looseMarshal, 2761
  - looseUnmarshal, 2761
  - MessagePullMarshaller, 2761
  - tightMarshal1, 2762
  - tightMarshal2, 2762
  - tightUnmarshal, 2763
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2816
  - ~NetworkBridgeFilterMarshaller, 2817
  - createObject, 2817
  - getDataStructureType, 2817
  - looseMarshal, 2817
  - looseUnmarshal, 2817
  - NetworkBridgeFilterMarshaller, 2817
  - tightMarshal1, 2818
  - tightMarshal2, 2818
  - tightUnmarshal, 2819
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2942
  - ~PartialCommandMarshaller, 2943
  - createObject, 2943
  - getDataStructureType, 2943
  - looseMarshal, 2943
  - looseUnmarshal, 2944
  - PartialCommandMarshaller, 2943
  - tightMarshal1, 2944
  - tightMarshal2, 2944
  - tightUnmarshal, 2945
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3060
  - ~ProducerAckMarshaller, 3061
  - createObject, 3061
  - getDataStructureType, 3061
  - looseMarshal, 3061
  - looseUnmarshal, 3061
  - ProducerAckMarshaller, 3061
  - tightMarshal1, 3062
  - tightMarshal2, 3062
  - tightUnmarshal, 3063
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3092
  - ~ProducerIdMarshaller, 3093
  - createObject, 3093
  - getDataStructureType, 3093
  - looseMarshal, 3093
  - looseUnmarshal, 3093
  - ProducerIdMarshaller, 3093
  - tightMarshal1, 3094
  - tightMarshal2, 3094
  - tightUnmarshal, 3095
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3109
  - ~ProducerInfoMarshaller, 3110
  - createObject, 3110
  - getDataStructureType, 3110
  - looseMarshal, 3110
  - looseUnmarshal, 3110
  - ProducerInfoMarshaller, 3110
  - tightMarshal1, 3111
  - tightMarshal2, 3111

- tightUnmarshal, 3112
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3406
  - 3210
  - ~RemoveInfoMarshaller, 3211
  - createObject, 3211
  - getDataStructureType, 3211
  - looseMarshal, 3211
  - looseUnmarshal, 3211
  - RemoveInfoMarshaller, 3211
  - tightMarshal1, 3212
  - tightMarshal2, 3212
  - tightUnmarshal, 3213
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3227
  - 3227
  - ~RemoveSubscriptionInfoMarshaller, 3228
  - createObject, 3228
  - getDataStructureType, 3228
  - looseMarshal, 3228
  - looseUnmarshal, 3228
  - RemoveSubscriptionInfoMarshaller, 3228
  - tightMarshal1, 3229
  - tightMarshal2, 3229
  - tightUnmarshal, 3230
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3258
  - 3258
  - ~ReplayCommandMarshaller, 3259
  - createObject, 3259
  - getDataStructureType, 3259
  - looseMarshal, 3259
  - looseUnmarshal, 3259
  - ReplayCommandMarshaller, 3259
  - tightMarshal1, 3260
  - tightMarshal2, 3260
  - tightUnmarshal, 3261
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3315
  - 3315
  - ~ResponseMarshaller, 3316
  - createObject, 3316
  - getDataStructureType, 3316
  - looseMarshal, 3316
  - looseUnmarshal, 3317
  - ResponseMarshaller, 3316
  - tightMarshal1, 3317
  - tightMarshal2, 3318
  - tightUnmarshal, 3318
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3403
  - 3403
  - ~SessionIdMarshaller, 3404
  - createObject, 3404
  - getDataStructureType, 3404
  - looseMarshal, 3404
  - looseUnmarshal, 3404
  - SessionIdMarshaller, 3404
  - tightMarshal1, 3405
- tightMarshal2, 3405
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3419
  - 3419
  - ~SessionInfoMarshaller, 3420
  - createObject, 3420
  - getDataStructureType, 3420
  - looseMarshal, 3420
  - looseUnmarshal, 3420
  - SessionInfoMarshaller, 3420
  - tightMarshal1, 3421
  - tightMarshal2, 3421
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3481
  - 3481
  - ~ShutdownInfoMarshaller, 3482
  - createObject, 3482
  - getDataStructureType, 3482
  - looseMarshal, 3482
  - looseUnmarshal, 3482
  - ShutdownInfoMarshaller, 3482
  - tightMarshal1, 3483
  - tightMarshal2, 3483
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3680
  - 3680
  - ~SubscriptionInfoMarshaller, 3681
  - createObject, 3681
  - getDataStructureType, 3681
  - looseMarshal, 3681
  - looseUnmarshal, 3681
  - SubscriptionInfoMarshaller, 3681
  - tightMarshal1, 3682
  - tightMarshal2, 3682
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3827
  - 3827
  - ~TransactionIdMarshaller, 3828
  - looseMarshal, 3828
  - looseUnmarshal, 3828
  - tightMarshal1, 3829
  - tightMarshal2, 3829
  - tightUnmarshal, 3830
  - TransactionIdMarshaller, 3828
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3855
  - 3855
  - ~TransactionInfoMarshaller, 3856
  - createObject, 3856
  - getDataStructureType, 3856
  - looseMarshal, 3856
  - looseUnmarshal, 3856
  - tightMarshal1, 3857
  - tightMarshal2, 3857
  - tightUnmarshal, 3858



- TransactionInfoMarshaller, 3856
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 4008
  - ~WireFormatInfoMarshaller, 4009
  - createObject, 4009
  - getDataStructureType, 4009
  - looseMarshal, 4009
  - looseUnmarshal, 4009
  - tightMarshal1, 4010
  - tightMarshal2, 4010
  - tightUnmarshal, 4011
  - WireFormatInfoMarshaller, 4009
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 4048
  - ~XATransactionIdMarshaller, 4049
  - createObject, 4049
  - getDataStructureType, 4049
  - looseMarshal, 4049
  - looseUnmarshal, 4049
  - tightMarshal1, 4050
  - tightMarshal2, 4050
  - tightUnmarshal, 4051
  - XATransactionIdMarshaller, 4049
- activemq::wireformat::openwire::marshal::v2, 113
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 221
  - ~ActiveMQBlobMessageMarshaller, 222
  - ActiveMQBlobMessageMarshaller, 222
  - createObject, 222
  - getDataStructureType, 222
  - looseMarshal, 222
  - looseUnmarshal, 222
  - tightMarshal1, 223
  - tightMarshal2, 223
  - tightUnmarshal, 224
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 269
  - ~ActiveMQBytesMessageMarshaller, 270
  - ActiveMQBytesMessageMarshaller, 270
  - createObject, 270
  - getDataStructureType, 270
  - looseMarshal, 270
  - looseUnmarshal, 270
  - tightMarshal1, 271
  - tightMarshal2, 271
  - tightUnmarshal, 272
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 349
  - ~ActiveMQDestinationMarshaller, 350
  - ActiveMQDestinationMarshaller, 350
  - looseMarshal, 350
  - looseUnmarshal, 350
  - tightMarshal1, 351
- tightMarshal2, 351
- activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 390
  - ~ActiveMQMapMessageMarshaller, 391
  - ActiveMQMapMessageMarshaller, 391
  - createObject, 391
  - getDataStructureType, 391
  - looseMarshal, 391
  - looseUnmarshal, 391
  - tightMarshal1, 392
  - tightMarshal2, 392
- activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 417
  - ~ActiveMQMessageMarshaller, 418
  - ActiveMQMessageMarshaller, 418
  - createObject, 418
  - getDataStructureType, 418
  - looseMarshal, 418
  - looseUnmarshal, 418
  - tightMarshal1, 419
  - tightMarshal2, 419
  - tightUnmarshal, 420
- activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 463
  - ~ActiveMQObjectMessageMarshaller, 463
  - ActiveMQObjectMessageMarshaller, 463
  - createObject, 463
  - getDataStructureType, 463
  - looseMarshal, 463
  - looseUnmarshal, 463
  - tightMarshal1, 464
  - tightMarshal2, 464
  - tightUnmarshal, 465
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller, 508
  - ~ActiveMQQueueMessageMarshaller, 508
  - ActiveMQQueueMessageMarshaller, 508
  - createObject, 508
  - getDataStructureType, 508
  - looseMarshal, 508
  - looseUnmarshal, 508
  - tightMarshal1, 509
  - tightMarshal2, 509
  - tightUnmarshal, 510
- activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 569
  - ~ActiveMQStreamMessageMarshaller, 569
  - ActiveMQStreamMessageMarshaller, 569
  - createObject, 569
  - getDataStructureType, 569
  - looseMarshal, 569
  - looseUnmarshal, 569

- tightMarshal1, 570
- tightMarshal2, 570
- tightUnmarshal, 571
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 596
  - ~ActiveMQTempDestinationMarshaller, 597
  - ActiveMQTempDestinationMarshaller, 597
  - looseMarshal, 597
  - looseUnmarshal, 597
  - tightMarshal1, 598
  - tightMarshal2, 598
  - tightUnmarshal, 599
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 625
  - ~ActiveMQTempQueueMarshaller, 626
  - ActiveMQTempQueueMarshaller, 626
  - createObject, 626
  - getDataStructureType, 626
  - looseMarshal, 626
  - looseUnmarshal, 626
  - tightMarshal1, 627
  - tightMarshal2, 627
  - tightUnmarshal, 628
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 654
  - ~ActiveMQTempTopicMarshaller, 655
  - ActiveMQTempTopicMarshaller, 655
  - createObject, 655
  - getDataStructureType, 655
  - looseMarshal, 655
  - looseUnmarshal, 655
  - tightMarshal1, 656
  - tightMarshal2, 656
  - tightUnmarshal, 657
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 687
  - ~ActiveMQTextMessageMarshaller, 688
  - ActiveMQTextMessageMarshaller, 688
  - createObject, 688
  - getDataStructureType, 688
  - looseMarshal, 688
  - looseUnmarshal, 688
  - tightMarshal1, 689
  - tightMarshal2, 689
  - tightUnmarshal, 690
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 715
  - ~ActiveMQTopicMarshaller, 716
  - ActiveMQTopicMarshaller, 716
  - createObject, 716
  - getDataStructureType, 716
  - looseMarshal, 716
  - looseUnmarshal, 716
- tightMarshal1, 717
- tightMarshal2, 717
- tightUnmarshal, 718
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 800
  - ~BaseCommandMarshaller, 801
  - BaseCommandMarshaller, 801
  - looseMarshal, 801
  - looseUnmarshal, 802
  - tightMarshal1, 803
  - tightMarshal2, 804
  - tightUnmarshal, 805
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 892
  - ~BrokerIdMarshaller, 893
  - BrokerIdMarshaller, 893
  - createObject, 893
  - getDataStructureType, 893
  - looseMarshal, 893
  - looseUnmarshal, 893
  - tightMarshal1, 894
  - tightMarshal2, 894
  - tightUnmarshal, 895
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 924
  - ~BrokerInfoMarshaller, 925
  - BrokerInfoMarshaller, 925
  - createObject, 925
  - getDataStructureType, 925
  - looseMarshal, 925
  - looseUnmarshal, 925
  - tightMarshal1, 926
  - tightMarshal2, 926
  - tightUnmarshal, 927
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1292
  - ~ConnectionControlMarshaller, 1293
  - ConnectionControlMarshaller, 1293
  - createObject, 1293
  - getDataStructureType, 1293
  - looseMarshal, 1293
  - looseUnmarshal, 1293
  - tightMarshal1, 1294
  - tightMarshal2, 1294
  - tightUnmarshal, 1295
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1300
  - ~ConnectionErrorMarshaller, 1301
  - ConnectionErrorMarshaller, 1301
  - createObject, 1301
  - getDataStructureType, 1301
  - looseMarshal, 1301
  - looseUnmarshal, 1301
  - tightMarshal1, 1302

- tightMarshal2, 1302
- tightUnmarshal, 1303
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1331
  - ~ConnectionIdMarshaller, 1332
  - ConnectionIdMarshaller, 1332
  - createObject, 1332
  - getDataStructureType, 1332
  - looseMarshal, 1332
  - looseUnmarshal, 1332
  - tightMarshal1, 1333
  - tightMarshal2, 1333
  - tightUnmarshal, 1334
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1361
  - ~ConnectionInfoMarshaller, 1362
  - ConnectionInfoMarshaller, 1362
  - createObject, 1362
  - getDataStructureType, 1362
  - looseMarshal, 1362
  - looseUnmarshal, 1362
  - tightMarshal1, 1363
  - tightMarshal2, 1363
  - tightUnmarshal, 1364
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1406
  - ~ConsumerControlMarshaller, 1407
  - ConsumerControlMarshaller, 1407
  - createObject, 1407
  - getDataStructureType, 1407
  - looseMarshal, 1407
  - looseUnmarshal, 1407
  - tightMarshal1, 1408
  - tightMarshal2, 1408
  - tightUnmarshal, 1409
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1435
  - ~ConsumerIdMarshaller, 1436
  - ConsumerIdMarshaller, 1436
  - createObject, 1436
  - getDataStructureType, 1436
  - looseMarshal, 1436
  - looseUnmarshal, 1436
  - tightMarshal1, 1437
  - tightMarshal2, 1437
  - tightUnmarshal, 1438
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1468
  - ~ConsumerInfoMarshaller, 1469
  - ConsumerInfoMarshaller, 1469
  - createObject, 1469
  - getDataStructureType, 1469
  - looseMarshal, 1469
  - looseUnmarshal, 1469
- tightMarshal1, 1470
- tightMarshal2, 1470
- tightUnmarshal, 1471
- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1497
  - ~ControlCommandMarshaller, 1498
  - ControlCommandMarshaller, 1498
  - createObject, 1498
  - getDataStructureType, 1498
  - looseMarshal, 1498
  - looseUnmarshal, 1498
  - tightMarshal1, 1499
  - tightMarshal2, 1499
- tightUnmarshal, 1500
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1531
  - ~DataArrayResponseMarshaller, 1532
  - createObject, 1532
  - DataArrayResponseMarshaller, 1532
  - getDataStructureType, 1532
  - looseMarshal, 1532
  - looseUnmarshal, 1533
  - tightMarshal1, 1533
  - tightMarshal2, 1533
- tightUnmarshal, 1534
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1594
  - ~DataResponseMarshaller, 1595
  - createObject, 1595
  - DataResponseMarshaller, 1595
  - getDataStructureType, 1595
  - looseMarshal, 1595
  - looseUnmarshal, 1596
  - tightMarshal1, 1596
  - tightMarshal2, 1596
- tightUnmarshal, 1597
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1732
  - ~DestinationInfoMarshaller, 1733
  - createObject, 1733
  - DestinationInfoMarshaller, 1733
  - getDataStructureType, 1733
  - looseMarshal, 1733
  - looseUnmarshal, 1733
  - tightMarshal1, 1734
  - tightMarshal2, 1734
- tightUnmarshal, 1735
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1766
  - ~DiscoveryEventMarshaller, 1767
  - createObject, 1767
  - DiscoveryEventMarshaller, 1767
  - getDataStructureType, 1767
  - looseMarshal, 1767

- looseUnmarshal, 1767
- tightMarshal1, 1768
- tightMarshal2, 1768
- tightUnmarshal, 1769
- activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1846
  - ~ExceptionResponseMarshaller, 1847
  - createObject, 1847
  - ExceptionResponseMarshaller, 1847
  - getDataStructureType, 1847
  - looseMarshal, 1847
  - looseUnmarshal, 1848
  - tightMarshal1, 1848
  - tightMarshal2, 1848
  - tightUnmarshal, 1849
- activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1944
  - ~FlushCommandMarshaller, 1945
  - createObject, 1945
  - FlushCommandMarshaller, 1945
  - getDataStructureType, 1945
  - looseMarshal, 1945
  - looseUnmarshal, 1945
  - tightMarshal1, 1946
  - tightMarshal2, 1946
  - tightUnmarshal, 1947
- activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2098
  - ~IntegerResponseMarshaller, 2099
  - createObject, 2099
  - getDataStructureType, 2099
  - IntegerResponseMarshaller, 2099
  - looseMarshal, 2099
  - looseUnmarshal, 2100
  - tightMarshal1, 2100
  - tightMarshal2, 2100
  - tightUnmarshal, 2101
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2163
  - ~JournalQueueAckMarshaller, 2164
  - createObject, 2164
  - getDataStructureType, 2164
  - JournalQueueAckMarshaller, 2164
  - looseMarshal, 2164
  - looseUnmarshal, 2164
  - tightMarshal1, 2165
  - tightMarshal2, 2165
  - tightUnmarshal, 2166
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2192
  - ~JournalTopicAckMarshaller, 2193
  - createObject, 2193
  - getDataStructureType, 2193
  - JournalTopicAckMarshaller, 2193
- looseMarshal, 2193
- looseUnmarshal, 2193
- tightMarshal1, 2194
- tightMarshal2, 2194
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2215
  - ~JournalTraceMarshaller, 2216
  - createObject, 2216
  - getDataStructureType, 2216
  - JournalTraceMarshaller, 2216
  - looseMarshal, 2216
  - looseUnmarshal, 2216
  - tightMarshal1, 2217
  - tightMarshal2, 2217
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2246
  - ~JournalTransactionMarshaller, 2247
  - createObject, 2247
  - getDataStructureType, 2247
  - JournalTransactionMarshaller, 2247
  - looseMarshal, 2247
  - looseUnmarshal, 2247
  - tightMarshal1, 2248
  - tightMarshal2, 2248
- activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2273
  - ~KeepAliveInfoMarshaller, 2274
  - createObject, 2274
  - getDataStructureType, 2274
  - KeepAliveInfoMarshaller, 2274
  - looseMarshal, 2274
  - looseUnmarshal, 2274
  - tightMarshal1, 2275
  - tightMarshal2, 2275
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2312
  - ~LastPartialCommandMarshaller, 2313
  - createObject, 2313
  - getDataStructureType, 2313
  - LastPartialCommandMarshaller, 2313
  - looseMarshal, 2313
  - looseUnmarshal, 2314
  - tightMarshal1, 2314
  - tightMarshal2, 2314
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2355
  - ~LocalTransactionIdMarshaller, 2356
  - createObject, 2356
  - getDataStructureType, 2356

- LocalTransactionIdMarshaller, 2356
- looseMarshal, 2356
- looseUnmarshal, 2356
- tightMarshal1, 2357
- tightMarshal2, 2357
- tightUnmarshal, 2358
- activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2492
  - ~MarshallerFactory, 2492
  - configure, 2492
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2569
  - ~MessageAckMarshaller, 2570
  - createObject, 2570
  - getDataStructureType, 2570
  - looseMarshal, 2570
  - looseUnmarshal, 2570
  - MessageAckMarshaller, 2570
  - tightMarshal1, 2571
  - tightMarshal2, 2571
  - tightUnmarshal, 2572
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2606
  - ~MessageDispatchMarshaller, 2607
  - createObject, 2607
  - getDataStructureType, 2607
  - looseMarshal, 2607
  - looseUnmarshal, 2607
  - MessageDispatchMarshaller, 2607
  - tightMarshal1, 2608
  - tightMarshal2, 2608
  - tightUnmarshal, 2609
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2639
  - ~MessageDispatchNotificationMarshaller, 2640
  - createObject, 2640
  - getDataStructureType, 2640
  - looseMarshal, 2640
  - looseUnmarshal, 2640
  - MessageDispatchNotificationMarshaller, 2640
  - tightMarshal1, 2641
  - tightMarshal2, 2641
  - tightUnmarshal, 2642
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2668
  - ~MessageIdMarshaller, 2669
  - createObject, 2669
  - getDataStructureType, 2669
  - looseMarshal, 2669
  - looseUnmarshal, 2669
  - MessageIdMarshaller, 2669
  - tightMarshal1, 2670
  - tightMarshal2, 2670
  - tightUnmarshal, 2671
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2744
  - ~MessagePullMarshaller, 2745
  - createObject, 2745
  - getDataStructureType, 2745
  - looseMarshal, 2745
  - looseUnmarshal, 2745
  - MessagePullMarshaller, 2745
  - tightMarshal1, 2746
  - tightMarshal2, 2746
  - tightUnmarshal, 2747
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2796
  - ~NetworkBridgeFilterMarshaller, 2797
  - createObject, 2797
  - getDataStructureType, 2797
  - looseMarshal, 2797
  - looseUnmarshal, 2797
  - NetworkBridgeFilterMarshaller, 2797
  - tightMarshal1, 2798
  - tightMarshal2, 2798
  - tightUnmarshal, 2799
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2926
  - ~PartialCommandMarshaller, 2927
  - createObject, 2927
  - getDataStructureType, 2927
  - looseMarshal, 2927
  - looseUnmarshal, 2928
  - PartialCommandMarshaller, 2927
  - tightMarshal1, 2928
  - tightMarshal2, 2928
  - tightUnmarshal, 2929
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 3041
  - ~ProducerAckMarshaller, 3041
  - createObject, 3041
  - getDataStructureType, 3041
  - looseMarshal, 3041
  - looseUnmarshal, 3041
  - ProducerAckMarshaller, 3041
  - tightMarshal1, 3042
  - tightMarshal2, 3042

- tightUnmarshal, 3043
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3072
  - ~ProducerIdMarshaller, 3073
  - createObject, 3073
  - getDataStructureType, 3073
  - looseMarshal, 3073
  - looseUnmarshal, 3073
  - ProducerIdMarshaller, 3073
  - tightMarshal1, 3074
  - tightMarshal2, 3074
  - tightUnmarshal, 3075
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3105
  - ~ProducerInfoMarshaller, 3106
  - createObject, 3106
  - getDataStructureType, 3106
  - looseMarshal, 3106
  - looseUnmarshal, 3106
  - ProducerInfoMarshaller, 3106
  - tightMarshal1, 3107
  - tightMarshal2, 3107
  - tightUnmarshal, 3108
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3198
  - ~RemoveInfoMarshaller, 3199
  - createObject, 3199
  - getDataStructureType, 3199
  - looseMarshal, 3199
  - looseUnmarshal, 3199
  - RemoveInfoMarshaller, 3199
  - tightMarshal1, 3200
  - tightMarshal2, 3200
  - tightUnmarshal, 3201
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3235
  - ~RemoveSubscriptionInfoMarshaller, 3236
  - createObject, 3236
  - getDataStructureType, 3236
  - looseMarshal, 3236
  - looseUnmarshal, 3236
  - RemoveSubscriptionInfoMarshaller, 3236
  - tightMarshal1, 3237
  - tightMarshal2, 3237
  - tightUnmarshal, 3238
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3262
  - ~ReplayCommandMarshaller, 3263
  - createObject, 3263
  - getDataStructureType, 3263
  - looseMarshal, 3263
  - looseUnmarshal, 3263
  - ReplayCommandMarshaller, 3263
  - tightMarshal1, 3264
- tightMarshal2, 3264
- tightUnmarshal, 3265
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3300
  - ~ResponseMarshaller, 3301
  - createObject, 3301
  - getDataStructureType, 3301
  - looseMarshal, 3301
  - looseUnmarshal, 3302
  - ResponseMarshaller, 3301
  - tightMarshal1, 3302
  - tightMarshal2, 3303
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3383
  - ~SessionIdMarshaller, 3384
  - createObject, 3384
  - getDataStructureType, 3384
  - looseMarshal, 3384
  - looseUnmarshal, 3384
  - SessionIdMarshaller, 3384
  - tightMarshal1, 3385
  - tightMarshal2, 3385
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3427
  - ~SessionInfoMarshaller, 3428
  - createObject, 3428
  - getDataStructureType, 3428
  - looseMarshal, 3428
  - looseUnmarshal, 3428
  - SessionInfoMarshaller, 3428
  - tightMarshal1, 3429
  - tightMarshal2, 3429
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3477
  - ~ShutdownInfoMarshaller, 3478
  - createObject, 3478
  - getDataStructureType, 3478
  - looseMarshal, 3478
  - looseUnmarshal, 3478
  - ShutdownInfoMarshaller, 3478
  - tightMarshal1, 3479
  - tightMarshal2, 3479
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3696
  - ~SubscriptionInfoMarshaller, 3697
  - createObject, 3697
  - getDataStructureType, 3697
  - looseMarshal, 3697
  - looseUnmarshal, 3697
  - SubscriptionInfoMarshaller, 3697

- tightMarshal1, 3698
  - tightMarshal2, 3698
  - tightUnmarshal, 3699
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 212
  - 3831
  - ~TransactionIdMarshaller, 3832
  - looseMarshal, 3832
  - looseUnmarshal, 3832
  - tightMarshal1, 3833
  - tightMarshal2, 3833
  - tightUnmarshal, 3834
  - TransactionIdMarshaller, 3832
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 251
  - 3871
  - ~TransactionInfoMarshaller, 3872
  - createObject, 3872
  - getDataStructureType, 3872
  - looseMarshal, 3872
  - looseUnmarshal, 3872
  - tightMarshal1, 3873
  - tightMarshal2, 3873
  - tightUnmarshal, 3874
  - TransactionInfoMarshaller, 3872
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 336
  - 4000
  - ~WireFormatInfoMarshaller, 4001
  - createObject, 4001
  - getDataStructureType, 4001
  - looseMarshal, 4001
  - looseUnmarshal, 4001
  - tightMarshal1, 4002
  - tightMarshal2, 4002
  - tightUnmarshal, 4003
  - WireFormatInfoMarshaller, 4001
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 401
  - 4040
  - ~XATransactionIdMarshaller, 4041
  - createObject, 4041
  - getDataStructureType, 4041
  - looseMarshal, 4041
  - looseUnmarshal, 4041
  - tightMarshal1, 4042
  - tightMarshal2, 4042
  - tightUnmarshal, 4043
  - XATransactionIdMarshaller, 4041
- activemq::wireformat::openwire::marshal::v3, 118
- activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 209
  - ~ActiveMQBlobMessageMarshaller, 210
  - ActiveMQBlobMessageMarshaller, 210
  - createObject, 210
  - getDataStructureType, 210
  - looseMarshal, 210
  - looseUnmarshal, 210
  - tightMarshal1, 211
  - tightMarshal2, 211
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 249
  - ~ActiveMQBytesMessageMarshaller, 250
  - ActiveMQBytesMessageMarshaller, 250
  - createObject, 250
  - getDataStructureType, 250
  - looseMarshal, 250
  - looseUnmarshal, 250
  - tightMarshal1, 251
  - tightMarshal2, 251
  - tightUnmarshal, 252
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 333
  - ~ActiveMQDestinationMarshaller, 334
  - ActiveMQDestinationMarshaller, 334
  - looseMarshal, 334
  - looseUnmarshal, 334
  - tightMarshal1, 335
  - tightMarshal2, 335
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 374
  - ~ActiveMQMapMessageMarshaller, 375
  - ActiveMQMapMessageMarshaller, 375
  - createObject, 375
  - getDataStructureType, 375
  - looseMarshal, 375
  - looseUnmarshal, 375
  - tightMarshal1, 376
  - tightMarshal2, 376
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 401
  - ~ActiveMQMessageMarshaller, 402
  - ActiveMQMessageMarshaller, 402
  - createObject, 402
  - getDataStructureType, 402
  - looseMarshal, 402
  - looseUnmarshal, 402
  - tightMarshal1, 403
  - tightMarshal2, 403
  - tightUnmarshal, 404
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 447
  - ~ActiveMQObjectMessageMarshaller, 447
  - ActiveMQObjectMessageMarshaller, 447
  - createObject, 447
  - getDataStructureType, 447
  - looseMarshal, 447
  - looseUnmarshal, 447

- tightMarshal1, 448
- tightMarshal2, 448
- tightUnmarshal, 449
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 491
  - ~ActiveMQQueueMarshaller, 492
  - ActiveMQQueueMarshaller, 492
  - createObject, 492
  - getDataStructureType, 492
  - looseMarshal, 492
  - looseUnmarshal, 492
  - tightMarshal1, 493
  - tightMarshal2, 493
  - tightUnmarshal, 494
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 552
  - ~ActiveMQStreamMessageMarshaller, 553
  - ActiveMQStreamMessageMarshaller, 553
  - createObject, 553
  - getDataStructureType, 553
  - looseMarshal, 553
  - looseUnmarshal, 553
  - tightMarshal1, 554
  - tightMarshal2, 554
  - tightUnmarshal, 555
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 580
  - ~ActiveMQTempDestinationMarshaller, 581
  - ActiveMQTempDestinationMarshaller, 581
  - looseMarshal, 581
  - looseUnmarshal, 581
  - tightMarshal1, 582
  - tightMarshal2, 582
  - tightUnmarshal, 583
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 609
  - ~ActiveMQTempQueueMarshaller, 610
  - ActiveMQTempQueueMarshaller, 610
  - createObject, 610
  - getDataStructureType, 610
  - looseMarshal, 610
  - looseUnmarshal, 610
  - tightMarshal1, 611
  - tightMarshal2, 611
  - tightUnmarshal, 612
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 638
  - ~ActiveMQTempTopicMarshaller, 639
  - ActiveMQTempTopicMarshaller, 639
  - createObject, 639
  - getDataStructureType, 639
  - looseMarshal, 639
  - looseUnmarshal, 639
- tightMarshal1, 640
- tightMarshal2, 640
- tightUnmarshal, 641
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 667
  - ~ActiveMQTextMessageMarshaller, 668
  - ActiveMQTextMessageMarshaller, 668
  - createObject, 668
  - getDataStructureType, 668
  - looseMarshal, 668
  - looseUnmarshal, 668
  - tightMarshal1, 669
  - tightMarshal2, 669
  - tightUnmarshal, 670
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 695
  - ~ActiveMQTopicMarshaller, 696
  - ActiveMQTopicMarshaller, 696
  - createObject, 696
  - getDataStructureType, 696
  - looseMarshal, 696
  - looseUnmarshal, 696
  - tightMarshal1, 697
  - tightMarshal2, 697
  - tightUnmarshal, 698
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 765
  - ~BaseCommandMarshaller, 766
  - BaseCommandMarshaller, 766
  - looseMarshal, 766
  - looseUnmarshal, 767
  - tightMarshal1, 768
  - tightMarshal2, 769
  - tightUnmarshal, 770
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 872
  - ~BrokerIdMarshaller, 873
  - BrokerIdMarshaller, 873
  - createObject, 873
  - getDataStructureType, 873
  - looseMarshal, 873
  - looseUnmarshal, 873
  - tightMarshal1, 874
  - tightMarshal2, 874
  - tightUnmarshal, 875
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 904
  - ~BrokerInfoMarshaller, 905
  - BrokerInfoMarshaller, 905
  - createObject, 905
  - getDataStructureType, 905
  - looseMarshal, 905
  - looseUnmarshal, 905
  - tightMarshal1, 906



- tightMarshal2, 906
- tightUnmarshal, 907
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1272
  - ~ConnectionControlMarshaller, 1273
  - ConnectionControlMarshaller, 1273
  - createObject, 1273
  - getDataStructureType, 1273
  - looseMarshal, 1273
  - looseUnmarshal, 1273
  - tightMarshal1, 1274
  - tightMarshal2, 1274
  - tightUnmarshal, 1275
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1304
  - ~ConnectionErrorMarshaller, 1305
  - ConnectionErrorMarshaller, 1305
  - createObject, 1305
  - getDataStructureType, 1305
  - looseMarshal, 1305
  - looseUnmarshal, 1305
  - tightMarshal1, 1306
  - tightMarshal2, 1306
  - tightUnmarshal, 1307
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1335
  - ~ConnectionIdMarshaller, 1336
  - ConnectionIdMarshaller, 1336
  - createObject, 1336
  - getDataStructureType, 1336
  - looseMarshal, 1336
  - looseUnmarshal, 1336
  - tightMarshal1, 1337
  - tightMarshal2, 1337
  - tightUnmarshal, 1338
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1365
  - ~ConnectionInfoMarshaller, 1366
  - ConnectionInfoMarshaller, 1366
  - createObject, 1366
  - getDataStructureType, 1366
  - looseMarshal, 1366
  - looseUnmarshal, 1366
  - tightMarshal1, 1367
  - tightMarshal2, 1367
  - tightUnmarshal, 1368
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1410
  - ~ConsumerControlMarshaller, 1411
  - ConsumerControlMarshaller, 1411
  - createObject, 1411
  - getDataStructureType, 1411
  - looseMarshal, 1411
  - looseUnmarshal, 1411
- tightMarshal1, 1412
- tightMarshal2, 1412
- tightUnmarshal, 1413
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1439
  - ~ConsumerIdMarshaller, 1440
  - ConsumerIdMarshaller, 1440
  - createObject, 1440
  - getDataStructureType, 1440
  - looseMarshal, 1440
  - looseUnmarshal, 1440
  - tightMarshal1, 1441
  - tightMarshal2, 1441
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1472
  - ~ConsumerInfoMarshaller, 1473
  - ConsumerInfoMarshaller, 1473
  - createObject, 1473
  - getDataStructureType, 1473
  - looseMarshal, 1473
  - looseUnmarshal, 1473
  - tightMarshal1, 1474
  - tightMarshal2, 1474
- activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1501
  - ~ControlCommandMarshaller, 1502
  - ControlCommandMarshaller, 1502
  - createObject, 1502
  - getDataStructureType, 1502
  - looseMarshal, 1502
  - looseUnmarshal, 1502
  - tightMarshal1, 1503
  - tightMarshal2, 1503
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1535
  - ~DataArrayResponseMarshaller, 1536
  - createObject, 1536
  - DataArrayResponseMarshaller, 1536
  - getDataStructureType, 1536
  - looseMarshal, 1536
  - looseUnmarshal, 1537
  - tightMarshal1, 1537
  - tightMarshal2, 1537
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1598
  - ~DataResponseMarshaller, 1599
  - createObject, 1599
  - DataResponseMarshaller, 1599
  - getDataStructureType, 1599
  - looseMarshal, 1599

- looseUnmarshal, 1600
- tightMarshal1, 1600
- tightMarshal2, 1600
- tightUnmarshal, 1601
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1736
  - ~DestinationInfoMarshaller, 1737
  - createObject, 1737
  - DestinationInfoMarshaller, 1737
  - getDataStructureType, 1737
  - looseMarshal, 1737
  - looseUnmarshal, 1737
  - tightMarshal1, 1738
  - tightMarshal2, 1738
  - tightUnmarshal, 1739
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1770
  - ~DiscoveryEventMarshaller, 1771
  - createObject, 1771
  - DiscoveryEventMarshaller, 1771
  - getDataStructureType, 1771
  - looseMarshal, 1771
  - looseUnmarshal, 1771
  - tightMarshal1, 1772
  - tightMarshal2, 1772
  - tightUnmarshal, 1773
- activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1850
  - ~ExceptionResponseMarshaller, 1851
  - createObject, 1851
  - ExceptionResponseMarshaller, 1851
  - getDataStructureType, 1851
  - looseMarshal, 1851
  - looseUnmarshal, 1852
  - tightMarshal1, 1852
  - tightMarshal2, 1852
  - tightUnmarshal, 1853
- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1948
  - ~FlushCommandMarshaller, 1949
  - createObject, 1949
  - FlushCommandMarshaller, 1949
  - getDataStructureType, 1949
  - looseMarshal, 1949
  - looseUnmarshal, 1949
  - tightMarshal1, 1950
  - tightMarshal2, 1950
  - tightUnmarshal, 1951
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2102
  - ~IntegerResponseMarshaller, 2103
  - createObject, 2103
  - getDataStructureType, 2103
  - IntegerResponseMarshaller, 2103
- looseMarshal, 2103
- looseUnmarshal, 2104
- tightMarshal1, 2104
- tightMarshal2, 2104
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2171
  - ~JournalQueueAckMarshaller, 2172
  - createObject, 2172
  - getDataStructureType, 2172
  - JournalQueueAckMarshaller, 2172
  - looseMarshal, 2172
  - looseUnmarshal, 2172
  - tightMarshal1, 2173
  - tightMarshal2, 2173
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2196
  - ~JournalTopicAckMarshaller, 2197
  - createObject, 2197
  - getDataStructureType, 2197
  - JournalTopicAckMarshaller, 2197
  - looseMarshal, 2197
  - looseUnmarshal, 2197
  - tightMarshal1, 2198
  - tightMarshal2, 2198
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2219
  - ~JournalTraceMarshaller, 2220
  - createObject, 2220
  - getDataStructureType, 2220
  - JournalTraceMarshaller, 2220
  - looseMarshal, 2220
  - looseUnmarshal, 2220
  - tightMarshal1, 2221
  - tightMarshal2, 2221
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2250
  - ~JournalTransactionMarshaller, 2251
  - createObject, 2251
  - getDataStructureType, 2251
  - JournalTransactionMarshaller, 2251
  - looseMarshal, 2251
  - looseUnmarshal, 2251
  - tightMarshal1, 2252
  - tightMarshal2, 2252
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2277
  - ~KeepAliveInfoMarshaller, 2278
  - createObject, 2278
  - getDataStructureType, 2278

- KeepAliveInfoMarshaller, 2278
- looseMarshal, 2278
- looseUnmarshal, 2278
- tightMarshal1, 2279
- tightMarshal2, 2279
- tightUnmarshal, 2280
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2308
  - ~LastPartialCommandMarshaller, 2309
  - createObject, 2309
  - getDataStructureType, 2309
  - LastPartialCommandMarshaller, 2309
  - looseMarshal, 2309
  - looseUnmarshal, 2310
  - tightMarshal1, 2310
  - tightMarshal2, 2310
  - tightUnmarshal, 2311
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2359
  - ~LocalTransactionIdMarshaller, 2360
  - createObject, 2360
  - getDataStructureType, 2360
  - LocalTransactionIdMarshaller, 2360
  - looseMarshal, 2360
  - looseUnmarshal, 2360
  - tightMarshal1, 2361
  - tightMarshal2, 2361
  - tightUnmarshal, 2362
- activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2488
  - ~MarshallerFactory, 2488
  - configure, 2488
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2573
  - ~MessageAckMarshaller, 2574
  - createObject, 2574
  - getDataStructureType, 2574
  - looseMarshal, 2574
  - looseUnmarshal, 2574
  - MessageAckMarshaller, 2574
  - tightMarshal1, 2575
  - tightMarshal2, 2575
  - tightUnmarshal, 2576
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2610
  - ~MessageDispatchMarshaller, 2611
  - createObject, 2611
  - getDataStructureType, 2611
  - looseMarshal, 2611
  - looseUnmarshal, 2611
  - MessageDispatchMarshaller, 2611
  - tightMarshal1, 2612
  - tightMarshal2, 2612
  - tightUnmarshal, 2613
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2643
  - ~MessageDispatchNotificationMarshaller, 2644
  - createObject, 2644
  - getDataStructureType, 2644
  - looseMarshal, 2644
  - looseUnmarshal, 2644
  - MessageDispatchNotificationMarshaller, 2644
  - tightMarshal1, 2645
  - tightMarshal2, 2645
  - tightUnmarshal, 2646
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2680
  - ~MessageIdMarshaller, 2681
  - createObject, 2681
  - getDataStructureType, 2681
  - looseMarshal, 2681
  - looseUnmarshal, 2681
  - MessageIdMarshaller, 2681
  - tightMarshal1, 2682
  - tightMarshal2, 2682
  - tightUnmarshal, 2683
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2698
  - ~MessageMarshaller, 2699
  - looseMarshal, 2699
  - looseUnmarshal, 2699
  - MessageMarshaller, 2699
  - tightMarshal1, 2700
  - tightMarshal2, 2700
  - tightUnmarshal, 2701
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2752
  - ~MessagePullMarshaller, 2753
  - createObject, 2753
  - getDataStructureType, 2753
  - looseMarshal, 2753
  - looseUnmarshal, 2753
  - MessagePullMarshaller, 2753
  - tightMarshal1, 2754
  - tightMarshal2, 2754
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2808
  - ~NetworkBridgeFilterMarshaller, 2809
  - createObject, 2809
  - getDataStructureType, 2809
  - looseMarshal, 2809
  - looseUnmarshal, 2809
  - NetworkBridgeFilterMarshaller, 2809
  - tightMarshal1, 2810
  - tightMarshal2, 2810

- tightUnmarshal, 2811
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2934
  - ~PartialCommandMarshaller, 2935
  - createObject, 2935
  - getDataStructureType, 2935
  - looseMarshal, 2935
  - looseUnmarshal, 2936
  - PartialCommandMarshaller, 2935
  - tightMarshal1, 2936
  - tightMarshal2, 2936
  - tightUnmarshal, 2937
- activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 3048
  - ~ProducerAckMarshaller, 3049
  - createObject, 3049
  - getDataStructureType, 3049
  - looseMarshal, 3049
  - looseUnmarshal, 3049
  - ProducerAckMarshaller, 3049
  - tightMarshal1, 3050
  - tightMarshal2, 3050
  - tightUnmarshal, 3051
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3080
  - ~ProducerIdMarshaller, 3081
  - createObject, 3081
  - getDataStructureType, 3081
  - looseMarshal, 3081
  - looseUnmarshal, 3081
  - ProducerIdMarshaller, 3081
  - tightMarshal1, 3082
  - tightMarshal2, 3082
  - tightUnmarshal, 3083
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3117
  - ~ProducerInfoMarshaller, 3118
  - createObject, 3118
  - getDataStructureType, 3118
  - looseMarshal, 3118
  - looseUnmarshal, 3118
  - ProducerInfoMarshaller, 3118
  - tightMarshal1, 3119
  - tightMarshal2, 3119
  - tightUnmarshal, 3120
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3206
  - ~RemoveInfoMarshaller, 3207
  - createObject, 3207
  - getDataStructureType, 3207
  - looseMarshal, 3207
  - looseUnmarshal, 3207
  - RemoveInfoMarshaller, 3207
  - tightMarshal1, 3208
- tightMarshal2, 3208
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3231
  - ~RemoveSubscriptionInfoMarshaller, 3232
  - createObject, 3232
  - getDataStructureType, 3232
  - looseMarshal, 3232
  - looseUnmarshal, 3232
  - RemoveSubscriptionInfoMarshaller, 3232
  - tightMarshal1, 3233
  - tightMarshal2, 3233
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3266
  - ~ReplayCommandMarshaller, 3267
  - createObject, 3267
  - getDataStructureType, 3267
  - looseMarshal, 3267
  - looseUnmarshal, 3267
  - ReplayCommandMarshaller, 3267
  - tightMarshal1, 3268
  - tightMarshal2, 3268
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3310
  - ~ResponseMarshaller, 3311
  - createObject, 3311
  - getDataStructureType, 3311
  - looseMarshal, 3311
  - looseUnmarshal, 3312
  - ResponseMarshaller, 3311
  - tightMarshal1, 3312
  - tightMarshal2, 3313
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3399
  - ~SessionIdMarshaller, 3400
  - createObject, 3400
  - getDataStructureType, 3400
  - looseMarshal, 3400
  - looseUnmarshal, 3400
  - SessionIdMarshaller, 3400
  - tightMarshal1, 3401
  - tightMarshal2, 3401
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3423
  - ~SessionInfoMarshaller, 3424
  - createObject, 3424
  - getDataStructureType, 3424
  - looseMarshal, 3424
  - looseUnmarshal, 3424
  - SessionInfoMarshaller, 3424

- tightMarshal1, 3425
- tightMarshal2, 3425
- tightUnmarshal, 3426
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3489
  - ~ShutdownInfoMarshaller, 3490
  - createObject, 3490
  - getDataStructureType, 3490
  - looseMarshal, 3490
  - looseUnmarshal, 3490
  - ShutdownInfoMarshaller, 3490
  - tightMarshal1, 3491
  - tightMarshal2, 3491
  - tightUnmarshal, 3492
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3676
  - ~SubscriptionInfoMarshaller, 3677
  - createObject, 3677
  - getDataStructureType, 3677
  - looseMarshal, 3677
  - looseUnmarshal, 3677
  - SubscriptionInfoMarshaller, 3677
  - tightMarshal1, 3678
  - tightMarshal2, 3678
  - tightUnmarshal, 3679
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3835
  - ~TransactionIdMarshaller, 3836
  - looseMarshal, 3836
  - looseUnmarshal, 3836
  - tightMarshal1, 3837
  - tightMarshal2, 3837
  - tightUnmarshal, 3838
  - TransactionIdMarshaller, 3836
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3859
  - ~TransactionInfoMarshaller, 3860
  - createObject, 3860
  - getDataStructureType, 3860
  - looseMarshal, 3860
  - looseUnmarshal, 3860
  - tightMarshal1, 3861
  - tightMarshal2, 3861
  - tightUnmarshal, 3862
  - TransactionInfoMarshaller, 3860
- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 4012
  - ~WireFormatInfoMarshaller, 4013
  - createObject, 4013
  - getDataStructureType, 4013
  - looseMarshal, 4013
  - looseUnmarshal, 4013
  - tightMarshal1, 4014
  - tightMarshal2, 4014
- tightUnmarshal, 4015
- WireFormatInfoMarshaller, 4013
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 4052
  - ~XATransactionIdMarshaller, 4053
  - createObject, 4053
  - getDataStructureType, 4053
  - looseMarshal, 4053
  - looseUnmarshal, 4053
  - tightMarshal1, 4054
  - tightMarshal2, 4054
  - tightUnmarshal, 4055
  - XATransactionIdMarshaller, 4053
- activemq::wireformat::openwire::marshal::v4, 4053
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 217
    - ~ActiveMQBlobMessageMarshaller, 218
    - ActiveMQBlobMessageMarshaller, 218
    - createObject, 218
    - getDataStructureType, 218
    - looseMarshal, 218
    - looseUnmarshal, 218
    - tightMarshal1, 219
    - tightMarshal2, 219
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 257
    - ~ActiveMQBytesMessageMarshaller, 258
    - ActiveMQBytesMessageMarshaller, 258
    - createObject, 258
    - getDataStructureType, 258
    - looseMarshal, 258
    - looseUnmarshal, 258
    - tightMarshal1, 259
    - tightMarshal2, 259
    - tightUnmarshal, 260
  - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 341
    - ~ActiveMQDestinationMarshaller, 342
    - ActiveMQDestinationMarshaller, 342
    - looseMarshal, 342
    - looseUnmarshal, 342
    - tightMarshal1, 343
    - tightMarshal2, 343
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 382
    - ~ActiveMQMapMessageMarshaller, 383
    - ActiveMQMapMessageMarshaller, 383
    - createObject, 383
    - getDataStructureType, 383
    - looseMarshal, 383
    - looseUnmarshal, 383

- tightMarshal1, 384
- tightMarshal2, 384
- tightUnmarshal, 385
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 409
  - ~ActiveMQMessageMarshaller, 410
  - ActiveMQMessageMarshaller, 410
  - createObject, 410
  - getDataStructureType, 410
  - looseMarshal, 410
  - looseUnmarshal, 410
  - tightMarshal1, 411
  - tightMarshal2, 411
  - tightUnmarshal, 412
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 454
  - ~ActiveMQObjectMessageMarshaller, 455
  - ActiveMQObjectMessageMarshaller, 455
  - createObject, 455
  - getDataStructureType, 455
  - looseMarshal, 455
  - looseUnmarshal, 455
  - tightMarshal1, 456
  - tightMarshal2, 456
  - tightUnmarshal, 457
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller, 499
  - ~ActiveMQQueueMarshaller, 500
  - ActiveMQQueueMarshaller, 500
  - createObject, 500
  - getDataStructureType, 500
  - looseMarshal, 500
  - looseUnmarshal, 500
  - tightMarshal1, 501
  - tightMarshal2, 501
  - tightUnmarshal, 502
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 560
  - ~ActiveMQStreamMessageMarshaller, 561
  - ActiveMQStreamMessageMarshaller, 561
  - createObject, 561
  - getDataStructureType, 561
  - looseMarshal, 561
  - looseUnmarshal, 561
  - tightMarshal1, 562
  - tightMarshal2, 562
  - tightUnmarshal, 563
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 588
  - ~ActiveMQTempDestinationMarshaller, 589
  - ActiveMQTempDestinationMarshaller, 589
  - looseMarshal, 589
  - looseUnmarshal, 589
- tightMarshal1, 590
- tightMarshal2, 590
- tightUnmarshal, 591
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 617
  - ~ActiveMQTempQueueMarshaller, 618
  - ActiveMQTempQueueMarshaller, 618
  - createObject, 618
  - getDataStructureType, 618
  - looseMarshal, 618
  - looseUnmarshal, 618
  - tightMarshal1, 619
  - tightMarshal2, 619
  - tightUnmarshal, 620
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 642
  - ~ActiveMQTempTopicMarshaller, 643
  - ActiveMQTempTopicMarshaller, 643
  - createObject, 643
  - getDataStructureType, 643
  - looseMarshal, 643
  - looseUnmarshal, 643
  - tightMarshal1, 644
  - tightMarshal2, 644
  - tightUnmarshal, 645
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 671
  - ~ActiveMQTextMessageMarshaller, 672
  - ActiveMQTextMessageMarshaller, 672
  - createObject, 672
  - getDataStructureType, 672
  - looseMarshal, 672
  - looseUnmarshal, 672
  - tightMarshal1, 673
  - tightMarshal2, 673
  - tightUnmarshal, 674
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 699
  - ~ActiveMQTopicMarshaller, 700
  - ActiveMQTopicMarshaller, 700
  - createObject, 700
  - getDataStructureType, 700
  - looseMarshal, 700
  - looseUnmarshal, 700
  - tightMarshal1, 701
  - tightMarshal2, 701
  - tightUnmarshal, 702
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 772
  - ~BaseCommandMarshaller, 773
  - BaseCommandMarshaller, 773
  - looseMarshal, 773
  - looseUnmarshal, 774
  - tightMarshal1, 775

- tightMarshal2, 776
- tightUnmarshal, 777
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 876
  - ~BrokerIdMarshaller, 877
  - BrokerIdMarshaller, 877
  - createObject, 877
  - getDataStructureType, 877
  - looseMarshal, 877
  - looseUnmarshal, 877
  - tightMarshal1, 878
  - tightMarshal2, 878
  - tightUnmarshal, 879
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 908
  - ~BrokerInfoMarshaller, 909
  - BrokerInfoMarshaller, 909
  - createObject, 909
  - getDataStructureType, 909
  - looseMarshal, 909
  - looseUnmarshal, 909
  - tightMarshal1, 910
  - tightMarshal2, 910
  - tightUnmarshal, 911
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1276
  - ~ConnectionControlMarshaller, 1277
  - ConnectionControlMarshaller, 1277
  - createObject, 1277
  - getDataStructureType, 1277
  - looseMarshal, 1277
  - looseUnmarshal, 1277
  - tightMarshal1, 1278
  - tightMarshal2, 1278
  - tightUnmarshal, 1279
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1308
  - ~ConnectionErrorMarshaller, 1309
  - ConnectionErrorMarshaller, 1309
  - createObject, 1309
  - getDataStructureType, 1309
  - looseMarshal, 1309
  - looseUnmarshal, 1309
  - tightMarshal1, 1310
  - tightMarshal2, 1310
  - tightUnmarshal, 1311
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1339
  - ~ConnectionIdMarshaller, 1340
  - ConnectionIdMarshaller, 1340
  - createObject, 1340
  - getDataStructureType, 1340
  - looseMarshal, 1340
  - looseUnmarshal, 1340
- tightMarshal1, 1341
- tightMarshal2, 1341
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1369
  - ~ConnectionInfoMarshaller, 1370
  - ConnectionInfoMarshaller, 1370
  - createObject, 1370
  - getDataStructureType, 1370
  - looseMarshal, 1370
  - looseUnmarshal, 1370
  - tightMarshal1, 1371
  - tightMarshal2, 1371
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1414
  - ~ConsumerControlMarshaller, 1415
  - ConsumerControlMarshaller, 1415
  - createObject, 1415
  - getDataStructureType, 1415
  - looseMarshal, 1415
  - looseUnmarshal, 1415
  - tightMarshal1, 1416
  - tightMarshal2, 1416
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1443
  - ~ConsumerIdMarshaller, 1444
  - ConsumerIdMarshaller, 1444
  - createObject, 1444
  - getDataStructureType, 1444
  - looseMarshal, 1444
  - looseUnmarshal, 1444
  - tightMarshal1, 1445
  - tightMarshal2, 1445
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1476
  - ~ConsumerInfoMarshaller, 1477
  - ConsumerInfoMarshaller, 1477
  - createObject, 1477
  - getDataStructureType, 1477
  - looseMarshal, 1477
  - looseUnmarshal, 1477
  - tightMarshal1, 1478
  - tightMarshal2, 1478
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1505
  - ~ControlCommandMarshaller, 1506
  - ControlCommandMarshaller, 1506
  - createObject, 1506
  - getDataStructureType, 1506
  - looseMarshal, 1506

- looseUnmarshal, 1506
- tightMarshal1, 1507
- tightMarshal2, 1507
- tightUnmarshal, 1508
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1539
  - ~DataArrayResponseMarshaller, 1540
  - createObject, 1540
  - DataArrayResponseMarshaller, 1540
  - getDataStructureType, 1540
  - looseMarshal, 1540
  - looseUnmarshal, 1541
  - tightMarshal1, 1541
  - tightMarshal2, 1541
  - tightUnmarshal, 1542
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1602
  - ~DataResponseMarshaller, 1603
  - createObject, 1603
  - DataResponseMarshaller, 1603
  - getDataStructureType, 1603
  - looseMarshal, 1603
  - looseUnmarshal, 1604
  - tightMarshal1, 1604
  - tightMarshal2, 1604
  - tightUnmarshal, 1605
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1740
  - ~DestinationInfoMarshaller, 1741
  - createObject, 1741
  - DestinationInfoMarshaller, 1741
  - getDataStructureType, 1741
  - looseMarshal, 1741
  - looseUnmarshal, 1741
  - tightMarshal1, 1742
  - tightMarshal2, 1742
  - tightUnmarshal, 1743
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1774
  - ~DiscoveryEventMarshaller, 1775
  - createObject, 1775
  - DiscoveryEventMarshaller, 1775
  - getDataStructureType, 1775
  - looseMarshal, 1775
  - looseUnmarshal, 1775
  - tightMarshal1, 1776
  - tightMarshal2, 1776
  - tightUnmarshal, 1777
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1858
  - ~ExceptionResponseMarshaller, 1859
  - createObject, 1859
  - ExceptionResponseMarshaller, 1859
  - getDataStructureType, 1859
- looseMarshal, 1859
- looseUnmarshal, 1860
- tightMarshal1, 1860
- tightMarshal2, 1860
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1952
  - ~FlushCommandMarshaller, 1953
  - createObject, 1953
  - FlushCommandMarshaller, 1953
  - getDataStructureType, 1953
  - looseMarshal, 1953
  - looseUnmarshal, 1953
  - tightMarshal1, 1954
  - tightMarshal2, 1954
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2106
  - ~IntegerResponseMarshaller, 2107
  - createObject, 2107
  - getDataStructureType, 2107
  - IntegerResponseMarshaller, 2107
  - looseMarshal, 2107
  - looseUnmarshal, 2108
  - tightMarshal1, 2108
  - tightMarshal2, 2108
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2175
  - ~JournalQueueAckMarshaller, 2176
  - createObject, 2176
  - getDataStructureType, 2176
  - JournalQueueAckMarshaller, 2176
  - looseMarshal, 2176
  - looseUnmarshal, 2176
  - tightMarshal1, 2177
  - tightMarshal2, 2177
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2204
  - ~JournalTopicAckMarshaller, 2205
  - createObject, 2205
  - getDataStructureType, 2205
  - JournalTopicAckMarshaller, 2205
  - looseMarshal, 2205
  - looseUnmarshal, 2205
  - tightMarshal1, 2206
  - tightMarshal2, 2206
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2227
  - ~JournalTraceMarshaller, 2228
  - createObject, 2228
  - getDataStructureType, 2228



- JournalTraceMarshaller, 2228
- looseMarshal, 2228
- looseUnmarshal, 2228
- tightMarshal1, 2229
- tightMarshal2, 2229
- tightUnmarshal, 2230
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2258
  - ~JournalTransactionMarshaller, 2259
  - createObject, 2259
  - getDataStructureType, 2259
  - JournalTransactionMarshaller, 2259
  - looseMarshal, 2259
  - looseUnmarshal, 2259
  - tightMarshal1, 2260
  - tightMarshal2, 2260
  - tightUnmarshal, 2261
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2281
  - ~KeepAliveInfoMarshaller, 2282
  - createObject, 2282
  - getDataStructureType, 2282
  - KeepAliveInfoMarshaller, 2282
  - looseMarshal, 2282
  - looseUnmarshal, 2282
  - tightMarshal1, 2283
  - tightMarshal2, 2283
  - tightUnmarshal, 2284
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2320
  - ~LastPartialCommandMarshaller, 2321
  - createObject, 2321
  - getDataStructureType, 2321
  - LastPartialCommandMarshaller, 2321
  - looseMarshal, 2321
  - looseUnmarshal, 2322
  - tightMarshal1, 2322
  - tightMarshal2, 2322
  - tightUnmarshal, 2323
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2367
  - ~LocalTransactionIdMarshaller, 2368
  - createObject, 2368
  - getDataStructureType, 2368
  - LocalTransactionIdMarshaller, 2368
  - looseMarshal, 2368
  - looseUnmarshal, 2368
  - tightMarshal1, 2369
  - tightMarshal2, 2369
  - tightUnmarshal, 2370
- activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2489
  - ~MarshallerFactory, 2489
  - configure, 2489
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2577
  - ~MessageAckMarshaller, 2578
  - createObject, 2578
  - getDataStructureType, 2578
  - looseMarshal, 2578
  - looseUnmarshal, 2578
  - MessageAckMarshaller, 2578
  - tightMarshal1, 2579
  - tightMarshal2, 2579
  - tightUnmarshal, 2580
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2618
  - ~MessageDispatchMarshaller, 2619
  - createObject, 2619
  - getDataStructureType, 2619
  - looseMarshal, 2619
  - looseUnmarshal, 2619
  - MessageDispatchMarshaller, 2619
  - tightMarshal1, 2620
  - tightMarshal2, 2620
  - tightUnmarshal, 2621
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2647
  - ~MessageDispatchNotificationMarshaller, 2648
  - createObject, 2648
  - getDataStructureType, 2648
  - looseMarshal, 2648
  - looseUnmarshal, 2648
  - MessageDispatchNotificationMarshaller, 2648
  - tightMarshal1, 2649
  - tightMarshal2, 2649
  - tightUnmarshal, 2650
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2672
  - ~MessageIdMarshaller, 2673
  - createObject, 2673
  - getDataStructureType, 2673
  - looseMarshal, 2673
  - looseUnmarshal, 2673
  - MessageIdMarshaller, 2673
  - tightMarshal1, 2674
  - tightMarshal2, 2674
  - tightUnmarshal, 2675
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2708
  - ~MessageMarshaller, 2709
  - looseMarshal, 2709
  - looseUnmarshal, 2709
  - MessageMarshaller, 2709
  - tightMarshal1, 2710
  - tightMarshal2, 2710

- tightUnmarshal, 2711
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2756
  - ~MessagePullMarshaller, 2757
  - createObject, 2757
  - getDataStructureType, 2757
  - looseMarshal, 2757
  - looseUnmarshal, 2757
  - MessagePullMarshaller, 2757
  - tightMarshal1, 2758
  - tightMarshal2, 2758
  - tightUnmarshal, 2759
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2812
  - ~NetworkBridgeFilterMarshaller, 2813
  - createObject, 2813
  - getDataStructureType, 2813
  - looseMarshal, 2813
  - looseUnmarshal, 2813
  - NetworkBridgeFilterMarshaller, 2813
  - tightMarshal1, 2814
  - tightMarshal2, 2814
  - tightUnmarshal, 2815
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2938
  - ~PartialCommandMarshaller, 2939
  - createObject, 2939
  - getDataStructureType, 2939
  - looseMarshal, 2939
  - looseUnmarshal, 2940
  - PartialCommandMarshaller, 2939
  - tightMarshal1, 2940
  - tightMarshal2, 2940
  - tightUnmarshal, 2941
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 3044
  - ~ProducerAckMarshaller, 3045
  - createObject, 3045
  - getDataStructureType, 3045
  - looseMarshal, 3045
  - looseUnmarshal, 3045
  - ProducerAckMarshaller, 3045
  - tightMarshal1, 3046
  - tightMarshal2, 3046
  - tightUnmarshal, 3047
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3076
  - ~ProducerIdMarshaller, 3077
  - createObject, 3077
  - getDataStructureType, 3077
  - looseMarshal, 3077
  - looseUnmarshal, 3077
  - ProducerIdMarshaller, 3077
  - tightMarshal1, 3078
- tightMarshal2, 3078
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3101
  - ~ProducerInfoMarshaller, 3102
  - createObject, 3102
  - getDataStructureType, 3102
  - looseMarshal, 3102
  - looseUnmarshal, 3102
  - ProducerInfoMarshaller, 3102
  - tightMarshal1, 3103
  - tightMarshal2, 3103
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3218
  - ~RemoveInfoMarshaller, 3219
  - createObject, 3219
  - getDataStructureType, 3219
  - looseMarshal, 3219
  - looseUnmarshal, 3219
  - RemoveInfoMarshaller, 3219
  - tightMarshal1, 3220
  - tightMarshal2, 3220
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3247
  - ~RemoveSubscriptionInfoMarshaller, 3248
  - createObject, 3248
  - getDataStructureType, 3248
  - looseMarshal, 3248
  - looseUnmarshal, 3248
  - RemoveSubscriptionInfoMarshaller, 3248
  - tightMarshal1, 3249
  - tightMarshal2, 3249
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3254
  - ~ReplayCommandMarshaller, 3255
  - createObject, 3255
  - getDataStructureType, 3255
  - looseMarshal, 3255
  - looseUnmarshal, 3255
  - ReplayCommandMarshaller, 3255
  - tightMarshal1, 3256
  - tightMarshal2, 3256
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3295
  - ~ResponseMarshaller, 3296
  - createObject, 3296
  - getDataStructureType, 3296
  - looseMarshal, 3296
  - looseUnmarshal, 3297
  - ResponseMarshaller, 3296

- tightMarshal1, 3297
- tightMarshal2, 3298
- tightUnmarshal, 3298
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3387
  - ~SessionIdMarshaller, 3388
  - createObject, 3388
  - getDataStructureType, 3388
  - looseMarshal, 3388
  - looseUnmarshal, 3388
  - SessionIdMarshaller, 3388
  - tightMarshal1, 3389
  - tightMarshal2, 3389
  - tightUnmarshal, 3390
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3431
  - ~SessionInfoMarshaller, 3432
  - createObject, 3432
  - getDataStructureType, 3432
  - looseMarshal, 3432
  - looseUnmarshal, 3432
  - SessionInfoMarshaller, 3432
  - tightMarshal1, 3433
  - tightMarshal2, 3433
  - tightUnmarshal, 3434
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3493
  - ~ShutdownInfoMarshaller, 3494
  - createObject, 3494
  - getDataStructureType, 3494
  - looseMarshal, 3494
  - looseUnmarshal, 3494
  - ShutdownInfoMarshaller, 3494
  - tightMarshal1, 3495
  - tightMarshal2, 3495
  - tightUnmarshal, 3496
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3688
  - ~SubscriptionInfoMarshaller, 3689
  - createObject, 3689
  - getDataStructureType, 3689
  - looseMarshal, 3689
  - looseUnmarshal, 3689
  - SubscriptionInfoMarshaller, 3689
  - tightMarshal1, 3690
  - tightMarshal2, 3690
  - tightUnmarshal, 3691
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3839
  - ~TransactionIdMarshaller, 3840
  - looseMarshal, 3840
  - looseUnmarshal, 3840
  - tightMarshal1, 3841
  - tightMarshal2, 3841
- tightUnmarshal, 3842
- TransactionIdMarshaller, 3840
- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3867
  - ~TransactionInfoMarshaller, 3868
  - createObject, 3868
  - getDataStructureType, 3868
  - looseMarshal, 3868
  - looseUnmarshal, 3868
  - tightMarshal1, 3869
  - tightMarshal2, 3869
  - tightUnmarshal, 3870
  - TransactionInfoMarshaller, 3868
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4004
  - ~WireFormatInfoMarshaller, 4005
  - createObject, 4005
  - getDataStructureType, 4005
  - looseMarshal, 4005
  - looseUnmarshal, 4005
  - tightMarshal1, 4006
  - tightMarshal2, 4006
  - tightUnmarshal, 4007
  - WireFormatInfoMarshaller, 4005
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 4044
  - ~XATransactionIdMarshaller, 4045
  - createObject, 4045
  - getDataStructureType, 4045
  - looseMarshal, 4045
  - looseUnmarshal, 4045
  - tightMarshal1, 4046
  - tightMarshal2, 4046
  - tightUnmarshal, 4047
  - XATransactionIdMarshaller, 4045
- activemq::wireformat::openwire::marshal::v5, 1128
  - Marshaller, 225
    - ~ActiveMQBlobMessageMarshaller, 226
    - ActiveMQBlobMessageMarshaller, 226
    - createObject, 226
    - getDataStructureType, 226
    - looseMarshal, 226
    - looseUnmarshal, 226
    - tightMarshal1, 227
    - tightMarshal2, 227
  - Marshaller, 228
    - ~ActiveMQBytesMessageMarshaller, 262
    - ActiveMQBytesMessageMarshaller, 262
    - createObject, 262
    - getDataStructureType, 262

- looseMarshal, 262
- looseUnmarshal, 262
- tightMarshal1, 263
- tightMarshal2, 263
- tightUnmarshal, 264
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 345
  - ~ActiveMQDestinationMarshaller, 346
  - ActiveMQDestinationMarshaller, 346
  - looseMarshal, 346
  - looseUnmarshal, 346
  - tightMarshal1, 347
  - tightMarshal2, 347
  - tightUnmarshal, 348
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 386
  - ~ActiveMQMapMessageMarshaller, 387
  - ActiveMQMapMessageMarshaller, 387
  - createObject, 387
  - getDataStructureType, 387
  - looseMarshal, 387
  - looseUnmarshal, 387
  - tightMarshal1, 388
  - tightMarshal2, 388
  - tightUnmarshal, 389
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 413
  - ~ActiveMQMessageMarshaller, 414
  - ActiveMQMessageMarshaller, 414
  - createObject, 414
  - getDataStructureType, 414
  - looseMarshal, 414
  - looseUnmarshal, 414
  - tightMarshal1, 415
  - tightMarshal2, 415
  - tightUnmarshal, 416
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 458
  - ~ActiveMQObjectMessageMarshaller, 459
  - ActiveMQObjectMessageMarshaller, 459
  - createObject, 459
  - getDataStructureType, 459
  - looseMarshal, 459
  - looseUnmarshal, 459
  - tightMarshal1, 460
  - tightMarshal2, 460
  - tightUnmarshal, 461
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 503
  - ~ActiveMQQueueMarshaller, 504
  - ActiveMQQueueMarshaller, 504
  - createObject, 504
  - getDataStructureType, 504
  - looseMarshal, 504
  - looseUnmarshal, 504
  - tightMarshal1, 505
  - tightMarshal2, 505
  - tightUnmarshal, 506
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 564
  - ~ActiveMQStreamMessageMarshaller, 565
  - ActiveMQStreamMessageMarshaller, 565
  - createObject, 565
  - getDataStructureType, 565
  - looseMarshal, 565
  - looseUnmarshal, 565
  - tightMarshal1, 566
  - tightMarshal2, 566
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 592
  - ~ActiveMQTempDestinationMarshaller, 593
  - ActiveMQTempDestinationMarshaller, 593
  - looseMarshal, 593
  - looseUnmarshal, 593
  - tightMarshal1, 594
  - tightMarshal2, 594
  - tightUnmarshal, 595
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 621
  - ~ActiveMQTempQueueMarshaller, 622
  - ActiveMQTempQueueMarshaller, 622
  - createObject, 622
  - getDataStructureType, 622
  - looseMarshal, 622
  - looseUnmarshal, 622
  - tightMarshal1, 623
  - tightMarshal2, 623
  - tightUnmarshal, 624
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 650
  - ~ActiveMQTempTopicMarshaller, 651
  - ActiveMQTempTopicMarshaller, 651
  - createObject, 651
  - getDataStructureType, 651
  - looseMarshal, 651
  - looseUnmarshal, 651
  - tightMarshal1, 652
  - tightMarshal2, 652
  - tightUnmarshal, 653
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 679
  - ~ActiveMQTextMessageMarshaller, 680
  - ActiveMQTextMessageMarshaller, 680
  - createObject, 680
  - getDataStructureType, 680
  - looseMarshal, 680

- looseUnmarshal, 680
- tightMarshal1, 681
- tightMarshal2, 681
- tightUnmarshal, 682
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 707
  - ~ActiveMQTopicMarshaller, 708
  - ActiveMQTopicMarshaller, 708
  - createObject, 708
  - getDataStructureType, 708
  - looseMarshal, 708
  - looseUnmarshal, 708
  - tightMarshal1, 709
  - tightMarshal2, 709
  - tightUnmarshal, 710
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 786
  - ~BaseCommandMarshaller, 787
  - BaseCommandMarshaller, 787
  - looseMarshal, 787
  - looseUnmarshal, 788
  - tightMarshal1, 789
  - tightMarshal2, 790
  - tightUnmarshal, 791
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 884
  - ~BrokerIdMarshaller, 885
  - BrokerIdMarshaller, 885
  - createObject, 885
  - getDataStructureType, 885
  - looseMarshal, 885
  - looseUnmarshal, 885
  - tightMarshal1, 886
  - tightMarshal2, 886
  - tightUnmarshal, 887
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 916
  - ~BrokerInfoMarshaller, 917
  - BrokerInfoMarshaller, 917
  - createObject, 917
  - getDataStructureType, 917
  - looseMarshal, 917
  - looseUnmarshal, 917
  - tightMarshal1, 918
  - tightMarshal2, 918
  - tightUnmarshal, 919
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1284
  - ~ConnectionControlMarshaller, 1285
  - ConnectionControlMarshaller, 1285
  - createObject, 1285
  - getDataStructureType, 1285
  - looseMarshal, 1285
  - looseUnmarshal, 1285
- tightMarshal1, 1286
- tightMarshal2, 1286
- tightUnmarshal, 1287
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1316
  - ~ConnectionErrorMarshaller, 1317
  - ConnectionErrorMarshaller, 1317
  - createObject, 1317
  - getDataStructureType, 1317
  - looseMarshal, 1317
  - looseUnmarshal, 1317
  - tightMarshal1, 1318
  - tightMarshal2, 1318
  - tightUnmarshal, 1319
- activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1347
  - ~ConnectionIdMarshaller, 1348
  - ConnectionIdMarshaller, 1348
  - createObject, 1348
  - getDataStructureType, 1348
  - looseMarshal, 1348
  - looseUnmarshal, 1348
  - tightMarshal1, 1349
  - tightMarshal2, 1349
- activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1377
  - ~ConnectionInfoMarshaller, 1378
  - ConnectionInfoMarshaller, 1378
  - createObject, 1378
  - getDataStructureType, 1378
  - looseMarshal, 1378
  - looseUnmarshal, 1378
  - tightMarshal1, 1379
  - tightMarshal2, 1379
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1422
  - ~ConsumerControlMarshaller, 1423
  - ConsumerControlMarshaller, 1423
  - createObject, 1423
  - getDataStructureType, 1423
  - looseMarshal, 1423
  - looseUnmarshal, 1423
  - tightMarshal1, 1424
  - tightMarshal2, 1424
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1451
  - ~ConsumerIdMarshaller, 1452
  - ConsumerIdMarshaller, 1452
  - createObject, 1452
  - getDataStructureType, 1452
  - looseMarshal, 1452

- looseUnmarshal, 1452
- tightMarshal1, 1453
- tightMarshal2, 1453
- tightUnmarshal, 1454
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1484
  - ~ConsumerInfoMarshaller, 1485
  - ConsumerInfoMarshaller, 1485
  - createObject, 1485
  - getDataStructureType, 1485
  - looseMarshal, 1485
  - looseUnmarshal, 1485
  - tightMarshal1, 1486
  - tightMarshal2, 1486
  - tightUnmarshal, 1487
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1513
  - ~ControlCommandMarshaller, 1514
  - ControlCommandMarshaller, 1514
  - createObject, 1514
  - getDataStructureType, 1514
  - looseMarshal, 1514
  - looseUnmarshal, 1514
  - tightMarshal1, 1515
  - tightMarshal2, 1515
  - tightUnmarshal, 1516
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1547
  - ~DataArrayResponseMarshaller, 1548
  - createObject, 1548
  - DataArrayResponseMarshaller, 1548
  - getDataStructureType, 1548
  - looseMarshal, 1548
  - looseUnmarshal, 1549
  - tightMarshal1, 1549
  - tightMarshal2, 1549
  - tightUnmarshal, 1550
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1586
  - ~DataResponseMarshaller, 1587
  - createObject, 1587
  - DataResponseMarshaller, 1587
  - getDataStructureType, 1587
  - looseMarshal, 1587
  - looseUnmarshal, 1588
  - tightMarshal1, 1588
  - tightMarshal2, 1588
  - tightUnmarshal, 1589
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1752
  - ~DestinationInfoMarshaller, 1753
  - createObject, 1753
  - DestinationInfoMarshaller, 1753
  - getDataStructureType, 1753
- looseMarshal, 1753
- looseUnmarshal, 1753
- tightMarshal1, 1754
- tightMarshal2, 1754
- tightUnmarshal, 1755
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1782
  - ~DiscoveryEventMarshaller, 1783
  - createObject, 1783
  - DiscoveryEventMarshaller, 1783
  - getDataStructureType, 1783
  - looseMarshal, 1783
  - looseUnmarshal, 1783
  - tightMarshal1, 1784
  - tightMarshal2, 1784
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1854
  - ~ExceptionResponseMarshaller, 1855
  - createObject, 1855
  - ExceptionResponseMarshaller, 1855
  - getDataStructureType, 1855
  - looseMarshal, 1855
  - looseUnmarshal, 1856
  - tightMarshal1, 1856
  - tightMarshal2, 1856
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1960
  - ~FlushCommandMarshaller, 1961
  - createObject, 1961
  - FlushCommandMarshaller, 1961
  - getDataStructureType, 1961
  - looseMarshal, 1961
  - looseUnmarshal, 1961
  - tightMarshal1, 1962
  - tightMarshal2, 1962
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2114
  - ~IntegerResponseMarshaller, 2115
  - createObject, 2115
  - getDataStructureType, 2115
  - IntegerResponseMarshaller, 2115
  - looseMarshal, 2115
  - looseUnmarshal, 2116
  - tightMarshal1, 2116
  - tightMarshal2, 2116
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2167
  - ~JournalQueueAckMarshaller, 2168
  - createObject, 2168
  - getDataStructureType, 2168

- JournalQueueAckMarshaller, 2168
- looseMarshal, 2168
- looseUnmarshal, 2168
- tightMarshal1, 2169
- tightMarshal2, 2169
- tightUnmarshal, 2170
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2188
  - ~JournalTopicAckMarshaller, 2189
  - createObject, 2189
  - getDataStructureType, 2189
  - JournalTopicAckMarshaller, 2189
  - looseMarshal, 2189
  - looseUnmarshal, 2189
  - tightMarshal1, 2190
  - tightMarshal2, 2190
  - tightUnmarshal, 2191
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2235
  - ~JournalTraceMarshaller, 2236
  - createObject, 2236
  - getDataStructureType, 2236
  - JournalTraceMarshaller, 2236
  - looseMarshal, 2236
  - looseUnmarshal, 2236
  - tightMarshal1, 2237
  - tightMarshal2, 2237
  - tightUnmarshal, 2238
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2254
  - ~JournalTransactionMarshaller, 2255
  - createObject, 2255
  - getDataStructureType, 2255
  - JournalTransactionMarshaller, 2255
  - looseMarshal, 2255
  - looseUnmarshal, 2255
  - tightMarshal1, 2256
  - tightMarshal2, 2256
  - tightUnmarshal, 2257
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2285
  - ~KeepAliveInfoMarshaller, 2286
  - createObject, 2286
  - getDataStructureType, 2286
  - KeepAliveInfoMarshaller, 2286
  - looseMarshal, 2286
  - looseUnmarshal, 2286
  - tightMarshal1, 2287
  - tightMarshal2, 2287
  - tightUnmarshal, 2288
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2316
  - ~LastPartialCommandMarshaller, 2317
  - createObject, 2317
  - getDataStructureType, 2317
  - LastPartialCommandMarshaller, 2317
  - looseMarshal, 2317
  - looseUnmarshal, 2318
  - tightMarshal1, 2318
  - tightMarshal2, 2318
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2363
  - ~LocalTransactionIdMarshaller, 2364
  - createObject, 2364
  - getDataStructureType, 2364
  - LocalTransactionIdMarshaller, 2364
  - looseMarshal, 2364
  - looseUnmarshal, 2364
  - tightMarshal1, 2365
  - tightMarshal2, 2365
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2490
  - ~MarshallerFactory, 2490
  - configure, 2490
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2585
  - ~MessageAckMarshaller, 2586
  - createObject, 2586
  - getDataStructureType, 2586
  - looseMarshal, 2586
  - MessageAckMarshaller, 2586
  - tightMarshal1, 2587
  - tightMarshal2, 2587
  - tightUnmarshal, 2588
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2614
  - ~MessageDispatchMarshaller, 2615
  - createObject, 2615
  - getDataStructureType, 2615
  - looseMarshal, 2615
  - MessageDispatchMarshaller, 2615
  - tightMarshal1, 2616
  - tightMarshal2, 2616
  - tightUnmarshal, 2617
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2655
  - ~MessageDispatchNotificationMarshaller, 2656
  - createObject, 2656
  - getDataStructureType, 2656
  - looseUnmarshal, 2656
  - MessageDispatchNotificationMarshaller, 2656

- tightMarshal1, 2657
- tightMarshal2, 2657
- tightUnmarshal, 2658
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2676
  - ~MessageIdMarshaller, 2677
  - createObject, 2677
  - getDataStructureType, 2677
  - looseMarshal, 2677
  - looseUnmarshal, 2677
  - MessageIdMarshaller, 2677
  - tightMarshal1, 2678
  - tightMarshal2, 2678
  - tightUnmarshal, 2679
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2693
  - ~MessageMarshaller, 2694
  - looseMarshal, 2694
  - looseUnmarshal, 2694
  - MessageMarshaller, 2694
  - tightMarshal1, 2695
  - tightMarshal2, 2695
  - tightUnmarshal, 2696
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2748
  - ~MessagePullMarshaller, 2749
  - createObject, 2749
  - getDataStructureType, 2749
  - looseMarshal, 2749
  - looseUnmarshal, 2749
  - MessagePullMarshaller, 2749
  - tightMarshal1, 2750
  - tightMarshal2, 2750
  - tightUnmarshal, 2751
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2804
  - ~NetworkBridgeFilterMarshaller, 2805
  - createObject, 2805
  - getDataStructureType, 2805
  - looseMarshal, 2805
  - looseUnmarshal, 2805
  - NetworkBridgeFilterMarshaller, 2805
  - tightMarshal1, 2806
  - tightMarshal2, 2806
  - tightUnmarshal, 2807
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2930
  - ~PartialCommandMarshaller, 2931
  - createObject, 2931
  - getDataStructureType, 2931
  - looseMarshal, 2931
  - looseUnmarshal, 2932
  - PartialCommandMarshaller, 2931
  - tightMarshal1, 2932
- tightMarshal2, 2932
- tightUnmarshal, 2933
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3052
  - ~ProducerAckMarshaller, 3053
  - createObject, 3053
  - getDataStructureType, 3053
  - looseMarshal, 3053
  - looseUnmarshal, 3053
  - ProducerAckMarshaller, 3053
  - tightMarshal1, 3054
  - tightMarshal2, 3054
  - tightUnmarshal, 3055
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3084
  - ~ProducerIdMarshaller, 3085
  - createObject, 3085
  - getDataStructureType, 3085
  - looseMarshal, 3085
  - looseUnmarshal, 3085
  - ProducerIdMarshaller, 3085
  - tightMarshal1, 3086
  - tightMarshal2, 3086
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3113
  - ~ProducerInfoMarshaller, 3114
  - createObject, 3114
  - getDataStructureType, 3114
  - looseMarshal, 3114
  - looseUnmarshal, 3114
  - ProducerInfoMarshaller, 3114
  - tightMarshal1, 3115
  - tightMarshal2, 3115
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3214
  - ~RemoveInfoMarshaller, 3215
  - createObject, 3215
  - getDataStructureType, 3215
  - looseMarshal, 3215
  - looseUnmarshal, 3215
  - RemoveInfoMarshaller, 3215
  - tightMarshal1, 3216
  - tightMarshal2, 3216
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3243
  - ~RemoveSubscriptionInfoMarshaller, 3244
  - createObject, 3244
  - getDataStructureType, 3244
  - looseMarshal, 3244
  - looseUnmarshal, 3244
  - RemoveSubscriptionInfoMarshaller, 3244



- tightMarshal1, 3245
- tightMarshal2, 3245
- tightUnmarshal, 3246
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshal1, 3274
- ~ReplayCommandMarshaller, 3275
- createObject, 3275
- getDataStructureType, 3275
- looseMarshal, 3275
- looseUnmarshal, 3275
- ReplayCommandMarshaller, 3275
- tightMarshal1, 3276
- tightMarshal2, 3276
- tightUnmarshal, 3277
- activemq::wireformat::openwire::marshal::v5::ResponseCommandMarshal, 3305
- ~ResponseMarshaller, 3306
- createObject, 3306
- getDataStructureType, 3306
- looseMarshal, 3306
- looseUnmarshal, 3307
- ResponseMarshaller, 3306
- tightMarshal1, 3307
- tightMarshal2, 3308
- tightUnmarshal, 3308
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3395
- ~SessionIdMarshaller, 3396
- createObject, 3396
- getDataStructureType, 3396
- looseMarshal, 3396
- looseUnmarshal, 3396
- SessionIdMarshaller, 3396
- tightMarshal1, 3397
- tightMarshal2, 3397
- tightUnmarshal, 3398
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3415
- ~SessionInfoMarshaller, 3416
- createObject, 3416
- getDataStructureType, 3416
- looseMarshal, 3416
- looseUnmarshal, 3416
- SessionInfoMarshaller, 3416
- tightMarshal1, 3417
- tightMarshal2, 3417
- tightUnmarshal, 3418
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3485
- ~ShutdownInfoMarshaller, 3486
- createObject, 3486
- getDataStructureType, 3486
- looseMarshal, 3486
- looseUnmarshal, 3486
- ShutdownInfoMarshaller, 3486
- tightMarshal1, 3487
- tightMarshal2, 3487
- tightUnmarshal, 3488
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3684
- ~SubscriptionInfoMarshaller, 3685
- createObject, 3685
- getDataStructureType, 3685
- looseMarshal, 3685
- looseUnmarshal, 3685
- SubscriptionInfoMarshaller, 3685
- tightMarshal1, 3686
- tightMarshal2, 3686
- tightUnmarshal, 3687
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3823
- ~TransactionIdMarshaller, 3824
- looseMarshal, 3824
- looseUnmarshal, 3824
- tightMarshal1, 3825
- tightMarshal2, 3825
- tightUnmarshal, 3826
- TransactionIdMarshaller, 3824
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3851
- ~TransactionInfoMarshaller, 3852
- createObject, 3852
- getDataStructureType, 3852
- looseMarshal, 3852
- looseUnmarshal, 3852
- tightMarshal1, 3853
- tightMarshal2, 3853
- tightUnmarshal, 3854
- TransactionInfoMarshaller, 3852
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3992
- ~WireFormatInfoMarshaller, 3993
- createObject, 3993
- getDataStructureType, 3993
- looseMarshal, 3993
- looseUnmarshal, 3993
- tightMarshal1, 3994
- tightMarshal2, 3994
- tightUnmarshal, 3995
- WireFormatInfoMarshaller, 3993
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 4056
- ~XATransactionIdMarshaller, 4057
- createObject, 4057
- getDataStructureType, 4057
- looseMarshal, 4057
- looseUnmarshal, 4057
- tightMarshal1, 4058

- tightMarshal2, 4058
- tightUnmarshal, 4059
- XATransactionIdMarshaller, 4057
- activemq::wireformat::openwire::marshal::v6, 133
- activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 229
  - ~ActiveMQBlobMessageMarshaller, 230
  - ActiveMQBlobMessageMarshaller, 230
  - createObject, 230
  - getDataStructureType, 230
  - looseMarshal, 230
  - looseUnmarshal, 230
  - tightMarshal1, 231
  - tightMarshal2, 231
  - tightUnmarshal, 232
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 265
  - ~ActiveMQBytesMessageMarshaller, 266
  - ActiveMQBytesMessageMarshaller, 266
  - createObject, 266
  - getDataStructureType, 266
  - looseMarshal, 266
  - looseUnmarshal, 266
  - tightMarshal1, 267
  - tightMarshal2, 267
  - tightUnmarshal, 268
- activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 353
  - ~ActiveMQDestinationMarshaller, 354
  - ActiveMQDestinationMarshaller, 354
  - looseMarshal, 354
  - looseUnmarshal, 354
  - tightMarshal1, 355
  - tightMarshal2, 355
  - tightUnmarshal, 356
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 394
  - ~ActiveMQMapMessageMarshaller, 395
  - ActiveMQMapMessageMarshaller, 395
  - createObject, 395
  - getDataStructureType, 395
  - looseMarshal, 395
  - looseUnmarshal, 395
  - tightMarshal1, 396
  - tightMarshal2, 396
  - tightUnmarshal, 397
- activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 421
  - ~ActiveMQMessageMarshaller, 422
  - ActiveMQMessageMarshaller, 422
  - createObject, 422
  - getDataStructureType, 422
  - looseMarshal, 422
  - looseUnmarshal, 422
  - tightMarshal1, 423
  - tightMarshal2, 423
  - tightUnmarshal, 424
- activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 467
  - ~ActiveMQObjectMessageMarshaller, 467
  - ActiveMQObjectMessageMarshaller, 467
  - createObject, 467
  - getDataStructureType, 467
  - looseMarshal, 467
  - looseUnmarshal, 467
  - tightMarshal1, 468
  - tightMarshal2, 468
  - tightUnmarshal, 469
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, 512
  - ~ActiveMQQueueMessageMarshaller, 512
  - ActiveMQQueueMessageMarshaller, 512
  - createObject, 512
  - getDataStructureType, 512
  - looseMarshal, 512
  - looseUnmarshal, 512
  - tightMarshal1, 513
  - tightMarshal2, 513
  - tightUnmarshal, 514
- activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 573
  - ~ActiveMQStreamMessageMarshaller, 573
  - ActiveMQStreamMessageMarshaller, 573
  - createObject, 573
  - getDataStructureType, 573
  - looseMarshal, 573
  - looseUnmarshal, 573
  - tightMarshal1, 574
  - tightMarshal2, 574
  - tightUnmarshal, 575
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 600
  - ~ActiveMQTempDestinationMarshaller, 601
  - ActiveMQTempDestinationMarshaller, 601
  - looseMarshal, 601
  - looseUnmarshal, 601
  - tightMarshal1, 602
  - tightMarshal2, 602
  - tightUnmarshal, 603
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 629
  - ~ActiveMQTempQueueMarshaller, 630
  - ActiveMQTempQueueMarshaller, 630
  - createObject, 630
  - getDataStructureType, 630
  - looseMarshal, 630

- looseUnmarshal, 630
- tightMarshal1, 631
- tightMarshal2, 631
- tightUnmarshal, 632
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 658
  - ~ActiveMQTempTopicMarshaller, 659
  - ActiveMQTempTopicMarshaller, 659
  - createObject, 659
  - getDataStructureType, 659
  - looseMarshal, 659
  - looseUnmarshal, 659
  - tightMarshal1, 660
  - tightMarshal2, 660
  - tightUnmarshal, 661
- activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 683
  - ~ActiveMQTextMessageMarshaller, 684
  - ActiveMQTextMessageMarshaller, 684
  - createObject, 684
  - getDataStructureType, 684
  - looseMarshal, 684
  - looseUnmarshal, 684
  - tightMarshal1, 685
  - tightMarshal2, 685
  - tightUnmarshal, 686
- activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller, 711
  - ~ActiveMQTopicMarshaller, 712
  - ActiveMQTopicMarshaller, 712
  - createObject, 712
  - getDataStructureType, 712
  - looseMarshal, 712
  - looseUnmarshal, 712
  - tightMarshal1, 713
  - tightMarshal2, 713
  - tightUnmarshal, 714
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 793
  - ~BaseCommandMarshaller, 794
  - BaseCommandMarshaller, 794
  - looseMarshal, 794
  - looseUnmarshal, 795
  - tightMarshal1, 796
  - tightMarshal2, 797
  - tightUnmarshal, 798
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 888
  - ~BrokerIdMarshaller, 889
  - BrokerIdMarshaller, 889
  - createObject, 889
  - getDataStructureType, 889
  - looseMarshal, 889
  - looseUnmarshal, 889
- tightMarshal1, 890
- tightMarshal2, 890
- tightUnmarshal, 891
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 920
  - ~BrokerInfoMarshaller, 921
  - BrokerInfoMarshaller, 921
  - createObject, 921
  - getDataStructureType, 921
  - looseMarshal, 921
  - looseUnmarshal, 921
  - tightMarshal1, 922
  - tightMarshal2, 922
  - tightUnmarshal, 923
- activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1288
  - ~ConnectionControlMarshaller, 1289
  - ConnectionControlMarshaller, 1289
  - createObject, 1289
  - getDataStructureType, 1289
  - looseMarshal, 1289
  - looseUnmarshal, 1289
  - tightMarshal1, 1290
  - tightMarshal2, 1290
  - tightUnmarshal, 1291
- activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1320
  - ~ConnectionErrorMarshaller, 1321
  - ConnectionErrorMarshaller, 1321
  - createObject, 1321
  - getDataStructureType, 1321
  - looseMarshal, 1321
  - looseUnmarshal, 1321
  - tightMarshal1, 1322
  - tightMarshal2, 1322
  - tightUnmarshal, 1323
- activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1351
  - ~ConnectionIdMarshaller, 1352
  - ConnectionIdMarshaller, 1352
  - createObject, 1352
  - getDataStructureType, 1352
  - looseMarshal, 1352
  - looseUnmarshal, 1352
  - tightMarshal1, 1353
  - tightMarshal2, 1353
  - tightUnmarshal, 1354
- activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1381
  - ~ConnectionInfoMarshaller, 1382
  - ConnectionInfoMarshaller, 1382
  - createObject, 1382
  - getDataStructureType, 1382
  - looseMarshal, 1382

- looseUnmarshal, 1382
- tightMarshal1, 1383
- tightMarshal2, 1383
- tightUnmarshal, 1384
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1426
  - ~ConsumerControlMarshaller, 1427
  - ConsumerControlMarshaller, 1427
  - createObject, 1427
  - getDataStructureType, 1427
  - looseMarshal, 1427
  - looseUnmarshal, 1427
  - tightMarshal1, 1428
  - tightMarshal2, 1428
  - tightUnmarshal, 1429
- activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1455
  - ~ConsumerIdMarshaller, 1456
  - ConsumerIdMarshaller, 1456
  - createObject, 1456
  - getDataStructureType, 1456
  - looseMarshal, 1456
  - looseUnmarshal, 1456
  - tightMarshal1, 1457
  - tightMarshal2, 1457
  - tightUnmarshal, 1458
- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1488
  - ~ConsumerInfoMarshaller, 1489
  - ConsumerInfoMarshaller, 1489
  - createObject, 1489
  - getDataStructureType, 1489
  - looseMarshal, 1489
  - looseUnmarshal, 1489
  - tightMarshal1, 1490
  - tightMarshal2, 1490
  - tightUnmarshal, 1491
- activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1517
  - ~ControlCommandMarshaller, 1518
  - ControlCommandMarshaller, 1518
  - createObject, 1518
  - getDataStructureType, 1518
  - looseMarshal, 1518
  - looseUnmarshal, 1518
  - tightMarshal1, 1519
  - tightMarshal2, 1519
  - tightUnmarshal, 1520
- activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1551
  - ~DataArrayResponseMarshaller, 1552
  - createObject, 1552
  - DataArrayResponseMarshaller, 1552
  - getDataStructureType, 1552
- looseMarshal, 1552
- looseUnmarshal, 1553
- tightMarshal1, 1553
- tightMarshal2, 1553
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1590
  - ~DataResponseMarshaller, 1591
  - createObject, 1591
  - DataResponseMarshaller, 1591
  - getDataStructureType, 1591
  - looseMarshal, 1591
  - looseUnmarshal, 1592
  - tightMarshal1, 1592
  - tightMarshal2, 1592
  - tightUnmarshal, 1593
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1748
  - ~DestinationInfoMarshaller, 1749
  - createObject, 1749
  - DestinationInfoMarshaller, 1749
  - getDataStructureType, 1749
  - looseMarshal, 1749
  - looseUnmarshal, 1749
  - tightMarshal1, 1750
  - tightMarshal2, 1750
  - tightUnmarshal, 1751
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1762
  - ~DiscoveryEventMarshaller, 1763
  - createObject, 1763
  - DiscoveryEventMarshaller, 1763
  - getDataStructureType, 1763
  - looseMarshal, 1763
  - looseUnmarshal, 1763
  - tightMarshal1, 1764
  - tightMarshal2, 1764
  - tightUnmarshal, 1765
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1842
  - ~ExceptionResponseMarshaller, 1843
  - createObject, 1843
  - ExceptionResponseMarshaller, 1843
  - getDataStructureType, 1843
  - looseMarshal, 1843
  - looseUnmarshal, 1844
  - tightMarshal1, 1844
  - tightMarshal2, 1844
  - tightUnmarshal, 1845
- activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1940
  - ~FlushCommandMarshaller, 1941
  - createObject, 1941
  - FlushCommandMarshaller, 1941

- getDataStructureType, 1941
  - looseMarshal, 1941
  - looseUnmarshal, 1941
  - tightMarshal1, 1942
  - tightMarshal2, 1942
  - tightUnmarshal, 1943
- activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2094
  - ~IntegerResponseMarshaller, 2095
  - createObject, 2095
  - getDataStructureType, 2095
  - IntegerResponseMarshaller, 2095
  - looseMarshal, 2095
  - looseUnmarshal, 2096
  - tightMarshal1, 2096
  - tightMarshal2, 2096
  - tightUnmarshal, 2097
- activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2159
  - ~JournalQueueAckMarshaller, 2160
  - createObject, 2160
  - getDataStructureType, 2160
  - JournalQueueAckMarshaller, 2160
  - looseMarshal, 2160
  - looseUnmarshal, 2160
  - tightMarshal1, 2161
  - tightMarshal2, 2161
  - tightUnmarshal, 2162
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2200
  - ~JournalTopicAckMarshaller, 2201
  - createObject, 2201
  - getDataStructureType, 2201
  - JournalTopicAckMarshaller, 2201
  - looseMarshal, 2201
  - looseUnmarshal, 2201
  - tightMarshal1, 2202
  - tightMarshal2, 2202
  - tightUnmarshal, 2203
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2223
  - ~JournalTraceMarshaller, 2224
  - createObject, 2224
  - getDataStructureType, 2224
  - JournalTraceMarshaller, 2224
  - looseMarshal, 2224
  - looseUnmarshal, 2224
  - tightMarshal1, 2225
  - tightMarshal2, 2225
  - tightUnmarshal, 2226
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2242
  - ~JournalTransactionMarshaller, 2243
  - createObject, 2243
  - getDataStructureType, 2243
  - JournalTransactionMarshaller, 2243
  - looseMarshal, 2243
  - looseUnmarshal, 2243
  - tightMarshal1, 2244
  - tightMarshal2, 2244
  - tightUnmarshal, 2245
- activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2269
  - ~KeepAliveInfoMarshaller, 2270
  - createObject, 2270
  - getDataStructureType, 2270
  - KeepAliveInfoMarshaller, 2270
  - looseMarshal, 2270
  - looseUnmarshal, 2270
  - tightMarshal1, 2271
  - tightMarshal2, 2271
- activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2304
  - ~LastPartialCommandMarshaller, 2305
  - createObject, 2305
  - getDataStructureType, 2305
  - LastPartialCommandMarshaller, 2305
  - looseMarshal, 2305
  - looseUnmarshal, 2306
  - tightMarshal1, 2306
  - tightMarshal2, 2306
- activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2351
  - ~LocalTransactionIdMarshaller, 2352
  - createObject, 2352
  - getDataStructureType, 2352
  - LocalTransactionIdMarshaller, 2352
  - looseMarshal, 2352
  - looseUnmarshal, 2352
  - tightMarshal1, 2353
  - tightMarshal2, 2353
- activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2487
  - ~MarshallerFactory, 2487
  - configure, 2487
- activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2565
  - ~MessageAckMarshaller, 2566
  - createObject, 2566
  - getDataStructureType, 2566
  - looseMarshal, 2566
  - looseUnmarshal, 2566
  - MessageAckMarshaller, 2566
  - tightMarshal1, 2567
  - tightMarshal2, 2567

- tightUnmarshal, 2568
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2626
  - ~MessageDispatchMarshaller, 2627
  - createObject, 2627
  - getDataStructureType, 2627
  - looseMarshal, 2627
  - looseUnmarshal, 2627
  - MessageDispatchMarshaller, 2627
  - tightMarshal1, 2628
  - tightMarshal2, 2628
  - tightUnmarshal, 2629
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2635
  - ~MessageDispatchNotificationMarshaller, 2636
  - createObject, 2636
  - getDataStructureType, 2636
  - looseMarshal, 2636
  - looseUnmarshal, 2636
  - MessageDispatchNotificationMarshaller, 2636
  - tightMarshal1, 2637
  - tightMarshal2, 2637
  - tightUnmarshal, 2638
- activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2684
  - ~MessageIdMarshaller, 2685
  - createObject, 2685
  - getDataStructureType, 2685
  - looseMarshal, 2685
  - looseUnmarshal, 2685
  - MessageIdMarshaller, 2685
  - tightMarshal1, 2686
  - tightMarshal2, 2686
  - tightUnmarshal, 2687
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2718
  - ~MessageMarshaller, 2719
  - looseMarshal, 2719
  - looseUnmarshal, 2719
  - MessageMarshaller, 2719
  - tightMarshal1, 2720
  - tightMarshal2, 2720
  - tightUnmarshal, 2721
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2764
  - ~MessagePullMarshaller, 2765
  - createObject, 2765
  - getDataStructureType, 2765
  - looseMarshal, 2765
  - looseUnmarshal, 2765
  - MessagePullMarshaller, 2765
  - tightMarshal1, 2766
- tightMarshal2, 2766
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2800
  - ~NetworkBridgeFilterMarshaller, 2801
  - createObject, 2801
  - getDataStructureType, 2801
  - looseMarshal, 2801
  - looseUnmarshal, 2801
  - NetworkBridgeFilterMarshaller, 2801
  - tightMarshal1, 2802
  - tightMarshal2, 2802
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2922
  - ~PartialCommandMarshaller, 2923
  - createObject, 2923
  - getDataStructureType, 2923
  - looseMarshal, 2923
  - looseUnmarshal, 2924
  - PartialCommandMarshaller, 2923
  - tightMarshal1, 2924
  - tightMarshal2, 2924
  - tightUnmarshal, 2925
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3057
  - ~ProducerAckMarshaller, 3057
  - createObject, 3057
  - getDataStructureType, 3057
  - looseMarshal, 3057
  - looseUnmarshal, 3057
  - ProducerAckMarshaller, 3057
  - tightMarshal1, 3058
  - tightMarshal2, 3058
  - tightUnmarshal, 3059
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3089
  - ~ProducerIdMarshaller, 3089
  - createObject, 3089
  - getDataStructureType, 3089
  - looseMarshal, 3089
  - looseUnmarshal, 3089
  - ProducerIdMarshaller, 3089
  - tightMarshal1, 3090
  - tightMarshal2, 3090
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3121
  - ~ProducerInfoMarshaller, 3122
  - createObject, 3122
  - getDataStructureType, 3122
  - looseMarshal, 3122
  - looseUnmarshal, 3122
  - ProducerInfoMarshaller, 3122

- tightMarshal1, 3123
- tightMarshal2, 3123
- tightUnmarshal, 3124
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3394
  - 3202
  - ~RemoveInfoMarshaller, 3203
  - createObject, 3203
  - getDataStructureType, 3203
  - looseMarshal, 3203
  - looseUnmarshal, 3203
  - RemoveInfoMarshaller, 3203
  - tightMarshal1, 3204
  - tightMarshal2, 3204
  - tightUnmarshal, 3205
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3239
  - ~RemoveSubscriptionInfoMarshaller, 3240
  - createObject, 3240
  - getDataStructureType, 3240
  - looseMarshal, 3240
  - looseUnmarshal, 3240
  - RemoveSubscriptionInfoMarshaller, 3240
  - tightMarshal1, 3241
  - tightMarshal2, 3241
  - tightUnmarshal, 3242
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3270
  - ~ReplayCommandMarshaller, 3271
  - createObject, 3271
  - getDataStructureType, 3271
  - looseMarshal, 3271
  - looseUnmarshal, 3271
  - ReplayCommandMarshaller, 3271
  - tightMarshal1, 3272
  - tightMarshal2, 3272
  - tightUnmarshal, 3273
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3320
  - ~ResponseMarshaller, 3321
  - createObject, 3321
  - getDataStructureType, 3321
  - looseMarshal, 3321
  - looseUnmarshal, 3322
  - ResponseMarshaller, 3321
  - tightMarshal1, 3322
  - tightMarshal2, 3323
  - tightUnmarshal, 3323
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3391
  - ~SessionIdMarshaller, 3392
  - createObject, 3392
  - getDataStructureType, 3392
  - looseMarshal, 3392
  - looseUnmarshal, 3392
- SessionIdMarshaller, 3392
- tightMarshal1, 3393
- tightMarshal2, 3393
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3411
  - ~SessionInfoMarshaller, 3412
  - createObject, 3412
  - getDataStructureType, 3412
  - looseMarshal, 3412
  - looseUnmarshal, 3412
  - SessionInfoMarshaller, 3412
  - tightMarshal1, 3413
  - tightMarshal2, 3413
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3473
  - ~ShutdownInfoMarshaller, 3474
  - createObject, 3474
  - getDataStructureType, 3474
  - looseMarshal, 3474
  - looseUnmarshal, 3474
  - ShutdownInfoMarshaller, 3474
  - tightMarshal1, 3475
  - tightMarshal2, 3475
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3692
  - ~SubscriptionInfoMarshaller, 3693
  - createObject, 3693
  - getDataStructureType, 3693
  - looseMarshal, 3693
  - looseUnmarshal, 3693
  - SubscriptionInfoMarshaller, 3693
  - tightMarshal1, 3694
  - tightMarshal2, 3694
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3843
  - ~TransactionIdMarshaller, 3844
  - looseMarshal, 3844
  - looseUnmarshal, 3844
  - tightMarshal1, 3845
  - tightMarshal2, 3845
  - tightUnmarshal, 3846
  - TransactionIdMarshaller, 3844
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3863
  - ~TransactionInfoMarshaller, 3864
  - createObject, 3864
  - getDataStructureType, 3864
  - looseMarshal, 3864
  - looseUnmarshal, 3864
  - tightMarshal1, 3865

- tightMarshal2, 3865
- tightUnmarshal, 3866
- TransactionInfoMarshaller, 3864
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3996
  - ~WireFormatInfoMarshaller, 3997
  - createObject, 3997
  - getDataStructureType, 3997
  - looseMarshal, 3997
  - looseUnmarshal, 3997
  - tightMarshal1, 3998
  - tightMarshal2, 3998
  - tightUnmarshal, 3999
  - WireFormatInfoMarshaller, 3997
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4036
  - ~XATransactionIdMarshaller, 4037
  - createObject, 4037
  - getDataStructureType, 4037
  - looseMarshal, 4037
  - looseUnmarshal, 4037
  - tightMarshal1, 4038
  - tightMarshal2, 4038
  - tightUnmarshal, 4039
  - XATransactionIdMarshaller, 4037
- activemq::wireformat::openwire::OpenWireFormat, 2887
  - ~OpenWireFormat, 2890
  - addMarshaller, 2890
  - createNegotiator, 2890
  - DEFAULT\_VERSION, 2898
  - destroyMarshalers, 2890
  - doUnmarshal, 2890
  - getCacheSize, 2891
  - getMaxInactivityDuration, 2891
  - getMaxInactivityDurationInitialDelay, 2891
  - getPreferredWireFormatInfo, 2891
  - getVersion, 2892
  - hasNegotiator, 2892
  - inReceive, 2892
  - isCacheEnabled, 2892
  - isSizePrefixDisabled, 2892
  - isStackTraceEnabled, 2893
  - isTcpNoDelayEnabled, 2893
  - isTightEncodingEnabled, 2893
  - looseMarshalNestedObject, 2893
  - looseUnmarshalNestedObject, 2894
  - marshal, 2894
  - NULL\_TYPE, 2898
  - OpenWireFormat, 2890
  - renegotiateWireFormat, 2894
  - setCacheEnabled, 2895
  - setCacheSize, 2895
  - setMaxInactivityDuration, 2895
  - setMaxInactivityDurationInitialDelay, 2895
  - setPreferredWireFormatInfo, 2895
  - setSizePrefixDisabled, 2896
  - setStackTraceEnabled, 2896
  - setTcpNoDelayEnabled, 2896
  - setTightEncodingEnabled, 2896
  - setVersion, 2896
  - tightMarshalNestedObject1, 2897
  - tightMarshalNestedObject2, 2897
  - tightUnmarshalNestedObject, 2897
  - unmarshal, 2898
- activemq::wireformat::openwire::OpenWireFormatFactory, 2899
  - ~OpenWireFormatFactory, 2899
  - createWireFormat, 2899
  - OpenWireFormatFactory, 2899
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2901
  - ~OpenWireFormatNegotiator, 2902
  - close, 2902
  - onCommand, 2902
  - oneway, 2902
  - onTransportException, 2903
  - OpenWireFormatNegotiator, 2901
  - request, 2903
  - start, 2904
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2905
  - ~OpenWireResponseBuilder, 2905
  - buildIncomingCommands, 2905
  - buildResponse, 2905
  - OpenWireResponseBuilder, 2905
- activemq::wireformat::openwire::utils, 138
- activemq::wireformat::openwire::utils::BooleanStream, 857
  - ~BooleanStream, 858
  - BooleanStream, 858
  - clear, 858
  - marshal, 858
  - marshalledSize, 858
  - readBoolean, 858
  - unmarshal, 858
  - writeBoolean, 859
- activemq::wireformat::openwire::utils::HexTable, 1984
  - ~HexTable, 1984
  - HexTable, 1984
  - operator[], 1984
  - size, 1985
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2732
  - ~MessagePropertyInterceptor, 2733



- getBooleanProperty, 2734
- getByteProperty, 2734
- getDoubleProperty, 2734
- getFloatProperty, 2734
- getIntProperty, 2735
- getLongProperty, 2735
- getShortProperty, 2735
- getStringProperty, 2735
- MessagePropertyInterceptor, 2733
- setBooleanProperty, 2736
- setByteProperty, 2736
- setDoubleProperty, 2736
- setFloatProperty, 2736
- setIntProperty, 2737
- setLongProperty, 2737
- setShortProperty, 2737
- setStringProperty, 2737
- activemq::wireformat::stomp, 139
- activemq::wireformat::stomp::StompCommandConstants, 3629
  - ABORT, 3631
  - ACK, 3631
  - ACK\_AUTO, 3631
  - ACK\_CLIENT, 3631
  - ACK\_INDIVIDUAL, 3631
  - BEGIN, 3631
  - BYTES, 3631
  - COMMIT, 3631
  - CONNECT, 3631
  - CONNECTED, 3631
  - DISCONNECT, 3631
  - ERROR\_CMD, 3631
  - HEADER\_ACK, 3631
  - HEADER\_CLIENT\_ID, 3631
  - HEADER\_CONSUMERPRIORITY, 3631
  - HEADER\_CONTENTLENGTH, 3631
  - HEADER\_CORRELATIONID, 3631
  - HEADER\_DESTINATION, 3631
  - HEADER\_DISPATCH\_ASYNC, 3631
  - HEADER\_EXCLUSIVE, 3631
  - HEADER\_EXPIRES, 3631
  - HEADER\_ID, 3631
  - HEADER\_JMSPRIORITY, 3631
  - HEADER\_LOGIN, 3631
  - HEADER\_MAXPENDINGMSGLIMIT, 3631
  - HEADER\_MESSAGE, 3631
  - HEADER\_MESSAGEID, 3631
  - HEADER\_NOLOCAL, 3631
  - HEADER\_OLDSUBSCRIPTIONNAME, 3631
  - HEADER\_PASSWORD, 3631
  - HEADER\_PERSISTENT, 3631
  - HEADER\_PREFETCHSIZE, 3631
  - HEADER\_RECEIPT\_REQUIRED, 3631
  - HEADER\_RECEIPTID, 3631
  - HEADER\_REDELIVERED, 3631
  - HEADER\_REDELIVERYCOUNT, 3631
  - HEADER\_REPLYTO, 3631
  - HEADER\_REQUESTID, 3631
  - HEADER\_RESPONSEID, 3631
  - HEADER\_RETROACTIVE, 3631
  - HEADER\_SELECTOR, 3631
  - HEADER\_SESSIONID, 3631
  - HEADER\_SUBSCRIPTION, 3631
  - HEADER\_SUBSCRIPTIONNAME, 3631
  - HEADER\_TIMESTAMP, 3631
  - HEADER\_TRANSACTIONID, 3631
  - HEADER\_TRANSFORMATION\_ERROR, 3631
  - HEADER\_TYPE, 3631
  - MESSAGE, 3631
  - QUEUE\_PREFIX, 3631
  - RECEIPT, 3631
  - SEND, 3631
  - SUBSCRIBE, 3631
  - TEMPQUEUE\_PREFIX, 3631
  - TEMPTOPIC\_PREFIX, 3631
  - TEXT, 3631
  - TOPIC\_PREFIX, 3631
  - UNSUBSCRIBE, 3631
- activemq::wireformat::stomp::StompFrame, 3633
  - ~StompFrame, 3634
  - clone, 3634
  - copy, 3634
  - fromStream, 3634
  - getBody, 3635
  - getBodyLength, 3635
  - getCommand, 3635
  - getProperties, 3635
  - getProperty, 3636
  - hasProperty, 3636
  - removeProperty, 3636
  - setBody, 3636
  - setCommand, 3636
  - setProperty, 3637
  - StompFrame, 3634
  - toStream, 3637
- activemq::wireformat::stomp::StompHelper, 3638
  - ~StompHelper, 3639
  - convertConsumerId, 3639
  - convertDestination, 3639, 3640
  - convertMessageId, 3640
  - convertProducerId, 3640, 3641
  - convertProperties, 3641

- convertTransactionId, 3641, 3642
- StompHelper, 3639
- activemq::wireformat::stomp::StompWireFormat, 3643
  - ~StompWireFormat, 3644
  - createNegotiator, 3644
  - getVersion, 3644
  - hasNegotiator, 3644
  - inReceive, 3644
  - marshal, 3645
  - setVersion, 3645
  - StompWireFormat, 3644
  - unmarshal, 3645
- activemq::wireformat::stomp::StompWireFormatFactory, 3647
  - ~StompWireFormatFactory, 3647
  - createWireFormat, 3647
  - StompWireFormatFactory, 3647
- activemq::wireformat::WireFormat, 3976
  - ~WireFormat, 3977
  - createNegotiator, 3977
  - getVersion, 3977
  - hasNegotiator, 3977
  - inReceive, 3978
  - marshal, 3978
  - setVersion, 3978
  - unmarshal, 3978
- activemq::wireformat::WireFormatFactory, 3980
  - ~WireFormatFactory, 3980
  - createWireFormat, 3980
- activemq::wireformat::WireFormatNegotiator, 4016
  - ~WireFormatNegotiator, 4016
  - WireFormatNegotiator, 4016
- activemq::wireformat::WireFormatRegistry, 4017
  - ~WireFormatRegistry, 4018
  - findFactory, 4018
  - getInstance, 4018
  - getWireFormatNames, 4018
  - registerFactory, 4018
  - unregisterFactory, 4019
- ActiveMQBlobMessage
  - activemq::commands::ActiveMQBlobMessage, 205
- ActiveMQBlobMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 214
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 222
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 210
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 218
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 226
- activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 230
- ActiveMQBytesMessage
  - activemq::commands::ActiveMQBytesMessage, 236
- ActiveMQBytesMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 254
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 270
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 250
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 258
  - activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 262
  - activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 266
- ActiveMQConnection
  - activemq::core::ActiveMQConnection, 278
- ActiveMQConnectionFactory
  - activemq::core::ActiveMQConnectionFactory, 294
- ActiveMQConnectionMetaData
  - activemq::core::ActiveMQConnectionMetaData, 304
- ActiveMQConsumer
  - activemq::core::ActiveMQConsumer, 312
- ActiveMQCPP
  - activemq::library::ActiveMQCPP, 320
- ActiveMQDestination
  - activemq::commands::ActiveMQDestination, 324
- ActiveMQDestinationMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 338
  - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 350
  - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 334
  - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 342
  - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 346
  - activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 354
- ActiveMQException
  - activemq::exceptions::ActiveMQException, 357
- ActiveMQMapMessage

activemq::commands::ActiveMQMapMessage, ActiveMQQueue  
 363  
 activemq::commands::ActiveMQQueue,  
 484  
 ActiveMQMapMessageMarshaller  
 379  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller,  
 379  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller,  
 391  
 ActiveMQQueueMarshaller  
 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller,  
 375  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,  
 496  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller,  
 383  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,  
 508  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller,  
 387  
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller,  
 492  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller,  
 395  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller,  
 500  
 ActiveMQMessage  
 activemq::commands::ActiveMQMessage,  
 398  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller,  
 504  
 ActiveMQMessageMarshaller  
 512  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,  
 406  
 ActiveMQSession  
 activemq::core::ActiveMQSession, 519  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller,  
 418  
 ActiveMQSessionExecutor  
 activemq::core::ActiveMQSession, 532  
 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller,  
 402  
 activemq::core::ActiveMQSessionExecutor,  
 534  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller,  
 410  
 ActiveMQStreamMessage  
 activemq::commands::ActiveMQStreamMessage,  
 414  
 ActiveMQMessageMarshaller,  
 340  
 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller,  
 422  
 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller,  
 557  
 ActiveMQMessageTemplate  
 activemq::commands::ActiveMQMessageTemplate,  
 428  
 569  
 ActiveMQObjectMessage  
 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller,  
 553  
 activemq::commands::ActiveMQObjectMessage,  
 444  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller,  
 561  
 ActiveMQObjectMessageMarshaller  
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,  
 451  
 565  
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller,  
 463  
 573  
 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,  
 447  
 ActiveMQTempDestination  
 activemq::commands::ActiveMQTempDestination,  
 455  
 577  
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller,  
 459  
 ActiveMQTempDestinationMarshaller  
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,  
 459  
 585  
 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller,  
 467  
 597  
 ActiveMQProducer  
 activemq::core::ActiveMQProducer, 471  
 581  
 ActiveMQProperties  
 activemq::util::ActiveMQProperties, 479  
 589

- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 593
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 601
- ActiveMQTempQueue
  - activemq::commands::ActiveMQTempQueue, 605
- ActiveMQTempQueueMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 614
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 626
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 610
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 618
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 622
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 630
- ActiveMQTempTopic
  - activemq::commands::ActiveMQTempTopic, 634
- ActiveMQTempTopicMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 647
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 655
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 639
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 643
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 651
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 659
- ActiveMQTextMessage
  - activemq::commands::ActiveMQTextMessage, 663
- ActiveMQTextMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 676
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 688
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 668
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 672
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 680
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 684
- ActiveMQTopic
  - activemq::commands::ActiveMQTopic, 692
- ActiveMQTopicMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 716
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 696
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 700
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 704
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 708
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller, 712
- ActiveMQTransactionContext
  - activemq::transport::failover::CloseTransportsTask, 1875
- ActiveMQTransactionContextMarshaller
  - activemq::transport::failover::FailoverTransport, 1875
- decaf::util::AbstractCollection, 182
- decaf::util::AbstractQueue, 195
- decaf::util::Collection, 1186
- decaf::util::ListIterator, 2345
- decaf::util::PriorityQueue, 3031
- decaf::util::StlList, 3594
- decaf::util::StlSet, 3625
- decaf::util::AbstractCollection, 182
- decaf::util::AbstractQueue, 195
- decaf::util::List, 2338
- decaf::util::StlList, 3594
- decaf::util::concurrent::atomic::AtomicInteger, 742
- addAsResource
- decaf::internal::net::Network, 2791
- addConsumer
- activemq::state::TransactionState, 3876
- activemq::cmsutil::ResourceLifecycleManager, 3876
- activemq::state::SessionState, 3438
- activemq::cmsutil::ResourceLifecycleManager, 3876
- addDispatcher
- activemq::core::ActiveMQConnection, 278
- addHandler

- decaf::util::logging::Logger, 2389
- additionalPredicate
  - activemq::commands::ConsumerInfo, 1466
- addLogger
  - decaf::util::logging::LogManager, 2409
- addMarshaller
  - activemq::wireformat::openwire::OpenWireFormatz\_stream\_s, 4062
  - 2890
- addMessageConsumer
  - activemq::cmsutil::ResourceLifecycleManageradvisory
  - 3283
- addMessageProducer
  - activemq::cmsutil::ResourceLifecycleManagerADVISORY\_PREFIX
  - 3283
- addNetworkResource
  - decaf::internal::net::Network, 2791
- addProducer
  - activemq::core::ActiveMQConnection, 278
  - activemq::core::ActiveMQSession, 519
  - activemq::state::SessionState, 3438
- addProducerState
  - activemq::state::TransactionState, 3876
- addPropertyChangeListener
  - decaf::util::logging::LogManager, 2409
- addResource
  - decaf::internal::util::ResourceLifecycleManagerafterRollback
  - 3281
- address
  - decaf::net::SocketImpl, 3536
- addressBytes
  - decaf::net::InetAddress, 2022
- addSession
  - activemq::cmsutil::ResourceLifecycleManager,
  - 3283
  - activemq::state::ConnectionState, 1390
- addSynchronization
  - activemq::core::ActiveMQTransactionContextallocate
  - 720
- addTask
  - activemq::threads::CompositeTaskRunner,
  - 1224
- addTempDestination
  - activemq::state::ConnectionState, 1390
- addTransactionState
  - activemq::state::ConnectionState, 1390
- addTransportListener
  - activemq::core::ActiveMQConnection, 278
- addURI
  - activemq::transport::CompositeTransport,
  - 1226
  - activemq::transport::failover::FailoverTransport,
  - 1876
  - activemq::transport::failover::URIPool,
  - 3943
- addURIs
  - activemq::transport::failover::URIPool,
  - 3943
- adjustMinimum
  - decaf::internal::util::TimerTaskHeap, 3805
- adler
  - Adler32
  - decaf::util::zip::Adler32, 723
- ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination,
  - 331
- after
  - decaf::util::Date, 1666
- afterCommit
  - activemq::core::Synchronization, 3715
- afterMarshal
  - activemq::commands::BaseDataStructure,
  - 831
  - activemq::wireformat::MarshalAware, 2484
- afterMessageIsConsumed
  - activemq::core::ActiveMQConsumer, 313
- afterRollback
  - activemq::core::Synchronization, 3715
- afterUnmarshal
  - activemq::commands::BaseDataStructure,
  - 831
  - activemq::commands::Message, 2520
  - activemq::commands::WireFormatInfo,
  - 3984
  - activemq::wireformat::MarshalAware, 2485
- ALL
  - decaf::util::logging::Level, 2335
- allocate
  - decaf::nio::ByteBuffer, 1036
  - decaf::nio::CharBuffer, 1122
  - decaf::nio::DoubleBuffer, 1811
  - decaf::nio::FloatBuffer, 1927
  - decaf::nio::IntBuffer, 2068
  - decaf::nio::LongBuffer, 2446
  - decaf::nio::ShortBuffer, 3461
- AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION
  - CMSEExceptionSupport.h, 4224
- AMQ\_CATCH\_EXCEPTION\_CONVERT
  - activemq/exceptions/ExceptionDefines.h,
  - 4170
- AMQ\_CATCH\_NOTHROW
  - activemq/exceptions/ExceptionDefines.h,
  - 4170
- AMQ\_CATCH\_RETHROW

- activemq/exceptions/ExceptionDefines.h, 4171
- AMQ\_CATCHALL\_NOTHROW
  - activemq/exceptions/ExceptionDefines.h, 4171
- AMQ\_CATCHALL\_THROW
  - activemq/exceptions/ExceptionDefines.h, 4171
- AMQCPP\_API
  - activemq/util/Config.h, 4227
- ANY\_CHILD
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1726
- ANY\_DESCENDENT
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1726
- anyBytes
  - decaf::net::InetAddress, 2022
- append
  - decaf::io::Writer, 4022, 4023
  - decaf::lang::Appendable, 726
  - decaf::nio::CharBuffer, 1123
- AprPool
  - decaf::internal::AprPool, 728
- array
  - decaf::internal::nio::ByteBuffer, 1003
  - decaf::internal::nio::CharArrayBuffer, 1113
  - decaf::internal::nio::DoubleArrayBuffer, 1803
  - decaf::internal::nio::FloatArrayBuffer, 1919
  - decaf::internal::nio::IntArrayBuffer, 2060
  - decaf::internal::nio::LongArrayBuffer, 2438
  - decaf::internal::nio::ShortArrayBuffer, 3453
  - decaf::nio::ByteBuffer, 1037
  - decaf::nio::CharBuffer, 1124
  - decaf::nio::DoubleBuffer, 1812
  - decaf::nio::FloatBuffer, 1927
  - decaf::nio::IntBuffer, 2068
  - decaf::nio::LongBuffer, 2447
  - decaf::nio::ShortBuffer, 3461
- arraycopy
  - decaf::lang::System, 3728, 3729
- arrayOffset
  - decaf::internal::nio::ByteBuffer, 1004
  - decaf::internal::nio::CharArrayBuffer, 1113
  - decaf::internal::nio::DoubleArrayBuffer, 1804
  - decaf::internal::nio::FloatArrayBuffer, 1920
  - decaf::internal::nio::IntArrayBuffer, 2061
  - decaf::internal::nio::LongArrayBuffer, 2439
  - decaf::internal::nio::ShortArrayBuffer, 3454
  - decaf::nio::ByteBuffer, 1037
- decaf::nio::CharBuffer, 1124
- decaf::nio::DoubleBuffer, 1812
- decaf::nio::FloatBuffer, 1928
- decaf::nio::IntBuffer, 2069
- decaf::nio::LongBuffer, 2447
- decaf::nio::ShortBuffer, 3462
- ArrayPointer
  - decaf::lang::ArrayPointer, 732, 733
- arrival
  - activemq::commands::Message, 2532
- asCharBuffer
  - decaf::internal::nio::ByteBuffer, 1004
  - decaf::nio::ByteBuffer, 1037
- asciiToModifiedUtf8
  - activemq/util::MarshallingSupport, 2494
- asDoubleBuffer
  - decaf::internal::nio::ByteBuffer, 1004
  - decaf::nio::ByteBuffer, 1038
- asFloatBuffer
  - decaf::internal::nio::ByteBuffer, 1005
  - decaf::nio::ByteBuffer, 1038
- asIntBuffer
  - decaf::internal::nio::ByteBuffer, 1005
  - decaf::nio::ByteBuffer, 1038
- asLongBuffer
  - decaf::internal::nio::ByteBuffer, 1005
  - decaf::nio::ByteBuffer, 1039
- asReadOnlyBuffer
  - decaf::internal::nio::ByteBuffer, 1006
  - decaf::internal::nio::CharArrayBuffer, 1114
  - decaf::internal::nio::DoubleArrayBuffer, 1804
  - decaf::internal::nio::FloatArrayBuffer, 1920
  - decaf::internal::nio::IntArrayBuffer, 2061
  - decaf::internal::nio::LongArrayBuffer, 2439
  - decaf::internal::nio::ShortArrayBuffer, 3454
  - decaf::nio::ByteBuffer, 1039
  - decaf::nio::CharBuffer, 1125
  - decaf::nio::DoubleBuffer, 1812
  - decaf::nio::FloatBuffer, 1928
  - decaf::nio::IntBuffer, 2069
  - decaf::nio::LongBuffer, 2447
  - decaf::nio::ShortBuffer, 3462
- Assert
  - zutil.h, 4745
- asShortBuffer
  - decaf::internal::nio::ByteBuffer, 1006
  - decaf::nio::ByteBuffer, 1039
- AsyncSignalReadErrorTask
  - activemq::transport::inactivity::InactivityMonitor, 2007
- AsyncWriteTask

- activemq::transport::inactivity::InactivityMonitor, 2007
- atEOF
  - decaf::util::zip::InflaterInputStream, 2041
- AtomicBoolean
  - decaf::util::concurrent::atomic::AtomicBoolean, 738
- AtomicInteger
  - decaf::util::concurrent::atomic::AtomicInteger, 742
- AtomicRefCounter
  - decaf::util::concurrent::atomic::AtomicRefCounter, 747
- AtomicReference
  - decaf::util::concurrent::atomic::AtomicReference, 750
- AUTO\_ACKNOWLEDGE
  - cms::Session, 3368
- avail\_in
  - z\_stream\_s, 4062
- avail\_out
  - z\_stream\_s, 4062
- available
  - decaf::internal::io::StandardInputStream, 3582
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2862
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
  - decaf::internal::net::tcp::TcpSocket, 3741
  - decaf::internal::net::tcp::TcpSocketInputStream, 3748
  - decaf::io::BlockingByteArrayInputStream, 840
  - decaf::io::BufferedInputStream, 936
  - decaf::io::ByteArrayInputStream, 1023
  - decaf::io::FilterInputStream, 1896
  - decaf::io::InputStream, 2045
  - decaf::io::PushbackInputStream, 3143
  - decaf::net::SocketImpl, 3531
  - decaf::util::zip::InflaterInputStream, 2038
- availablePermits
  - decaf::util::concurrent::Semaphore, 3345
- availableProcessors
  - decaf::lang::System, 3729
- await
  - decaf::util::concurrent::CountDownLatch, 1522, 1523
  - decaf::util::concurrent::locks::Condition, 1251
- awaitNanos
  - decaf::util::concurrent::locks::Condition, 1252
- awaitTermination
- awaitUntil
  - decaf::util::concurrent::locks::Condition, 1254
- back
  - decaf::util::StlQueue, 3617
- inflate\_state, 2025
- BackupTransport
  - activemq::transport::failover::BackupTransport, 752
  - activemq::transport::failover::BackupTransportPool, 757
- BackupTransportPool
  - activemq::transport::failover::BackupTransportPool, 756
- BAD
  - inflate.h, 4733
- base\_dist
  - trees.h, 4736
- base\_length
  - trees.h, 4736
- BaseCommand
  - activemq::commands::BaseCommand, 759
- BaseCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::BaseCommandM, 780
  - activemq::wireformat::openwire::marshal::v2::BaseCommandM, 801
  - activemq::wireformat::openwire::marshal::v3::BaseCommandM, 766
  - activemq::wireformat::openwire::marshal::v4::BaseCommandM, 773
  - activemq::wireformat::openwire::marshal::v5::BaseCommandM, 787
  - activemq::wireformat::openwire::marshal::v6::BaseCommandM, 794
- before
  - decaf::util::Date, 1666
- beforeEnd
  - activemq::core::Synchronization, 3715
- beforeMarshal
  - activemq::commands::ActiveMQMapMessage, 363
  - activemq::commands::ActiveMQTextMessage, 663
  - activemq::commands::BaseDataStructure, 831
  - activemq::commands::Message, 2520

- activemq::commands::WireFormatInfo, 3985
- activemq::wireformat::MarshalAware, 2485
- beforeMessageIsConsumed
- activemq::core::ActiveMQConsumer, 313
- beforeUnmarshal
- activemq::commands::BaseDataStructure, 831
- activemq::wireformat::MarshalAware, 2485
- BEGIN
- activemq::wireformat::stomp::StompCommandConstants, 3631
- begin
- activemq::core::ActiveMQTransactionContext, 720
- BEST\_COMPRESSION
- decaf::util::zip::Deflater, 1714
- BEST\_SPEED
- decaf::util::zip::Deflater, 1714
- bi\_buf
- internal\_state, 2120
- bi\_valid
- internal\_state, 2120
- BIG\_STRING\_TYPE
- activemq::util::PrimitiveValueNode, 3016
- BINARY\_MIME\_TYPE
- activemq::commands::ActiveMQBlobMessage, 208
- bind
- decaf::internal::net::tcp::TcpSocket, 3741
- decaf::net::ServerSocket, 3356
- decaf::net::Socket, 3508
- decaf::net::SocketImpl, 3531
- BindException
- decaf::net::BindException, 836, 837
- bitCount
- decaf::lang::Integer, 2080
- decaf::lang::Long, 2423
- bits
- code, 1183
- inflate\_state, 2025
- BL\_CODES
- deflate.h, 4726
- bl\_count
- internal\_state, 2120
- bl\_desc
- internal\_state, 2120
- bl\_tree
- internal\_state, 2120
- block\_start
- internal\_state, 2120
- BLOCKED
- decaf::lang::Thread, 3768
- BlockingByteArrayInputStream
- decaf::io::BlockingByteArrayInputStream, 840
- Boolean
- decaf::lang::Boolean, 851
- BOOLEAN\_TYPE
- activemq::util::PrimitiveValueNode, 3015
- BooleanExpression
- activemq::commands::BooleanExpression, 855
- BooleanStream
- activemq::wireformat::openwire::utils::BooleanStream, 858
- booleanValue
- decaf::lang::Boolean, 851
- boolValue
- activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
- branchQualifier
- activemq::commands::XATransactionId, 4035
- BrokenBarrierException
- decaf::util::concurrent::BrokenBarrierException, 860, 861
- BrokerError
- activemq::commands::BrokerError, 864
- BrokerException
- activemq::exceptions::BrokerException, 867
- BrokerId
- activemq::commands::BrokerId, 870
- brokerId
- activemq::commands::BrokerInfo, 903
- BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 881
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 893
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 873
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 877
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 885
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 889
- BrokerInfo
- activemq::commands::BrokerInfo, 897
- BrokerInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 913
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 925
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 905



- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 909
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshallerUnderflowException, 957, 958
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 917
- activemq::transport::mock::ResponseBuilder, 3289
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 921
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2905
- brokerInTime
  - activemq::commands::Message, 2532
- brokerMasterConnector
  - activemq::commands::ConnectionInfo, 1360
- brokerName
  - activemq::commands::BrokerInfo, 903
  - activemq::commands::DiscoveryEvent, 1761
- brokerOutTime
  - activemq::commands::Message, 2532
- brokerPath
  - activemq::commands::ConnectionInfo, 1360
  - activemq::commands::ConsumerInfo, 1466
  - activemq::commands::DestinationInfo, 1731
  - activemq::commands::Message, 2532
  - activemq::commands::ProducerInfo, 3100
- brokerSequenceId
  - activemq::commands::MessageId, 2667
- brokerUploadUrl
  - activemq::commands::BrokerInfo, 903
- brokerURL
  - activemq::commands::BrokerInfo, 903
- Browser
  - activemq::core::ActiveMQQueueBrowser, 490
- browser
  - activemq::commands::ConsumerInfo, 1466
- buf
  - decaf::util::zip::DeflaterOutputStream, 1719
- buff
  - decaf::util::zip::InflaterInputStream, 2041
- Buffer
  - decaf::nio::Buffer, 930
- buffer
  - decaf::io::DataOutputStream, 1582
- BufferedInputStream
  - decaf::io::BufferedInputStream, 936
- BufferedOutputStream
  - decaf::io::BufferedOutputStream, 940
- BufferOverflowException
  - decaf::nio::BufferOverflowException, 954, 955
- BufferUnderflowException
- buildMessage
  - decaf::lang::Exception, 1834
- buildResponse
  - activemq::transport::mock::ResponseBuilder, 3290
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2905
- BUSY\_STATE
  - deflate.h, 4727
- Byte
  - decaf::lang::Byte, 961
  - zconf.h, 4738
- BYTE\_ARRAY\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- BYTE\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- ByteArrayAdapter
  - decaf::internal::util::ByteArrayAdapter, 973–976
- ByteArrayBuffer
  - decaf::internal::nio::ByteArrayBuffer, 1002, 1003
- ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 1022, 1023
- ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 1028
- byteArrayValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
- ByteBuffer
  - decaf::nio::ByteBuffer, 1036
- Bytcf
  - zconf.h, 4738
- BYTES
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- bytesToInt
  - decaf::net::InetAddress, 2018
- byteValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
  - decaf::lang::Byte, 962
  - decaf::lang::Character, 1102
  - decaf::lang::Double, 1790
  - decaf::lang::Float, 1906
  - decaf::lang::Integer, 2080

- decaf::lang::Long, 2423
- decaf::lang::Number, 2835
- decaf::lang::Short, 3442
- CachedConsumer
  - activemq::cmsutil::CachedConsumer, 1072
- CachedProducer
  - activemq::cmsutil::CachedProducer, 1076
- call
  - decaf::util::concurrent::Callable, 1082
- cancel
  - decaf::util::concurrent::Future, 1967
  - decaf::util::Timer, 3792
  - decaf::util::TimerTask, 3802
- CancellationException
  - decaf::util::concurrent::CancellationException, 1083, 1084
- capacity
  - decaf::nio::Buffer, 930
- cause
  - decaf::lang::Exception, 1837
- ceil
  - decaf::lang::Math, 2500
- CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1090, 1091
- CertificateException
  - decaf::security::cert::CertificateException, 1092, 1093
- CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1094, 1095
- CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1096, 1097
- CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1098, 1099
- CHAR\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- Character
  - decaf::lang::Character, 1102
- CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1111, 1112
- charAt
  - decaf::lang::CharSequence, 1135
  - decaf::lang::String, 3666
  - decaf::nio::CharBuffer, 1125
- CharBuffer
  - decaf::nio::CharBuffer, 1122
- charf
  - zconf.h, 4738
- charValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
- CHECK
  - inflate.h, 4733
- check
  - inflate\_state, 2025
- checkClosed
  - decaf::io::InputStreamReader, 2054
  - decaf::io::OutputStreamWriter, 2916
  - decaf::net::ServerSocket, 3357
  - decaf::net::Socket, 3509
- checkConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1154
- checkDestinationResolver
  - activemq::cmsutil::CmsDestinationAccessor, 1158
- CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1138
- CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1140
- checkMapIsUnmarshalled
  - activemq::commands::ActiveMQMapMessage, 363
- checkResult
  - decaf::internal::net::tcp::TcpSocket, 3741
- checkShutdown
  - activemq::state::ConnectionState, 1390
  - activemq::state::SessionState, 3438
  - activemq::state::TransactionState, 3876
- checkValidity
  - decaf::security::cert::X509Certificate, 4029
- ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1145, 1146
- ClassName
  - activemq::commands::BrokerError::StackTraceElement, 3578
- cleanup
  - decaf::internal::AprPool, 728
- clear
  - activemq::core::ActiveMQSessionExecutor, 534
  - activemq::core::MessageDispatchChannel, 2600
  - activemq::util::ActiveMQProperties, 479
  - activemq::util::PrimitiveValueNode, 3018
  - activemq::wireformat::openwire::utils::BooleanStream, 858
  - cms::CMSProperties, 1166
  - decaf::internal::util::ByteArrayAdapter, 976
  - decaf::nio::Buffer, 930
  - decaf::util::AbstractCollection, 183

- decaf::util::AbstractQueue, 196
- decaf::util::Collection, 1187
- decaf::util::concurrent::ConcurrentStlMap, 1237
- decaf::util::concurrent::SynchronousQueue, 3718
- decaf::util::Map, 2460
- decaf::util::PriorityQueue, 3031
- decaf::util::Properties, 3128
- decaf::util::StlList, 3596
- decaf::util::StlMap, 3606
- decaf::util::StlQueue, 3617
- decaf::util::StlSet, 3626
- clearBody
  - activemq::commands::ActiveMQBytesMessage, 236
  - activemq::commands::ActiveMQMapMessage, 363
  - activemq::commands::ActiveMQMessageTemplate, 428
  - activemq::commands::ActiveMQStreamMessage, 540
  - activemq::commands::ActiveMQTextMessage, 663
  - cms::Message, 2538
- clearMessagesInProgress
  - activemq::core::ActiveMQConsumer, 313
  - activemq::core::ActiveMQSession, 519
  - activemq::core::ActiveMQSessionExecutor, 534
- clearProperties
  - activemq::commands::ActiveMQMessageTemplate, 429
  - cms::Message, 2539
- clearProperty
  - decaf::lang::System, 3729
- CLIENT\_ACKNOWLEDGE
  - cms::Session, 3368
- clientId
  - activemq::commands::ConnectionInfo, 1360
  - activemq::commands::JournalTopicAck, 2187
  - activemq::commands::RemoveSubscriptionInfo, 3226
  - activemq::commands::SubscriptionInfo, 3675
- clientMaster
  - activemq::commands::ConnectionInfo, 1360
- clockSequence
  - decaf::util::UUID, 3971
- clone
  - activemq::commands::ActiveMQBlobMessage, 205
  - activemq::commands::ActiveMQBytesMessage, 236
  - activemq::commands::ActiveMQMapMessage, 363
  - activemq::commands::ActiveMQMessage, 398
  - activemq::commands::ActiveMQObjectMessage, 444
  - activemq::commands::ActiveMQQueue, 484
  - activemq::commands::ActiveMQStreamMessage, 540
  - activemq::commands::ActiveMQTempQueue, 605
  - activemq::commands::ActiveMQTempTopic, 634
  - activemq::commands::ActiveMQTextMessage, 663
  - activemq::commands::ActiveMQTopic, 692
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
  - activemq::core::PrefetchPolicy, 2977
  - activemq::core::RedeliveryPolicy, 3178
  - activemq::exceptions::ActiveMQException, 358
  - activemq::exceptions::BrokerException, 867
  - activemq::util::ActiveMQProperties, 479
  - activemq::wireformat::stomp::StompFrame, 3634
  - cms::BytesMessage, 1059
  - cms::CMSProperties, 1166
  - cms::Destination, 1724
  - cms::Message, 2539
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2845
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2873
  - decaf::io::EOFException, 1827
  - decaf::io::InterruptedIOException, 2129
  - decaf::io::IOException, 2144
  - decaf::io::UnsupportedEncodingException, 3915
  - decaf::io::UTFDataFormatException, 3968
  - decaf::lang::ArrayPointer, 733
  - decaf::lang::Exception, 1834
  - decaf::lang::exceptions::ClassCastException, 1147
  - decaf::lang::exceptions::IllegalArgumentException, 1993

- decaf::lang::exceptions::IllegalMonitorStateException, 1996
- decaf::lang::exceptions::IllegalStateException, 2000
- decaf::lang::exceptions::IllegalThreadStateException, 2003
- decaf::lang::exceptions::IndexOutOfBoundsException, 2010
- decaf::lang::exceptions::InterruptedException, 2126
- decaf::lang::exceptions::InvalidStateException, 2141
- decaf::lang::exceptions::NoSuchElementException, 2828
- decaf::lang::exceptions::NullPointerException, 2834
- decaf::lang::exceptions::NumberFormatException, 2840
- decaf::lang::exceptions::RuntimeException, 3330
- decaf::lang::exceptions::UnsupportedOperationException, 3918
- decaf::lang::Throwable, 3784
- decaf::net::BindException, 838
- decaf::net::ConnectException, 1261
- decaf::net::HttpRetryException, 1988
- decaf::net::MalformedURLException, 2458
- decaf::net::NoRouteToHostException, 2822
- decaf::net::PortUnreachableException, 2975
- decaf::net::ProtocolException, 3139
- decaf::net::SocketException, 3522
- decaf::net::SocketTimeoutException, 3544
- decaf::net::UnknownHostException, 3909
- decaf::net::UnknownServiceException, 3912
- decaf::net::URISyntaxException, 3950
- decaf::nio::BufferOverflowException, 956
- decaf::nio::BufferUnderflowException, 959
- decaf::nio::InvalidMarkException, 2137
- decaf::nio::ReadOnlyBufferException, 3171
- decaf::security::cert::CertificateEncodingException, 1091
- decaf::security::cert::CertificateException, 1093
- decaf::security::cert::CertificateExpiredException, 1095
- decaf::security::cert::CertificateNotYetValidException, 1097
- decaf::security::cert::CertificateParsingException, 1099
- decaf::security::GeneralSecurityException, 1973
- decaf::security::InvalidKeyException, 2134
- decaf::security::KeyException, 2297
- decaf::security::KeyManagementException, 2300
- decaf::security::NoSuchAlgorithmException, 2825
- decaf::security::NoSuchProviderException, 2831
- decaf::security::SignatureException, 3499
- decaf::util::concurrent::BrokenBarrierException, 862
- decaf::util::concurrent::CancellationException, 1085
- decaf::util::concurrent::ExecutionException, 1868
- decaf::util::concurrent::RejectedExecutionException, 3191
- decaf::util::concurrent::TimeoutException, 3789
- decaf::util::Properties, 3128
- decaf::util::zip::DataFormatException, 4057
- decaf::util::zip::ZipException, 4066
- cloneDataStructure
  - activemq::commands::ActiveMQBlobMessage, 205
  - activemq::commands::ActiveMQBytesMessage, 236
  - activemq::commands::ActiveMQDestination, 324
  - activemq::commands::ActiveMQMapMessage, 363
  - activemq::commands::ActiveMQMessage, 399
  - activemq::commands::ActiveMQObjectMessage, 444
  - activemq::commands::ActiveMQQueue, 484
  - activemq::commands::ActiveMQStreamMessage, 540
  - activemq::commands::ActiveMQTempDestination, 577
  - activemq::commands::ActiveMQTempQueue, 605
  - activemq::commands::ActiveMQTempTopic, 634
  - activemq::commands::ActiveMQTextMessage, 664
  - activemq::commands::ActiveMQTopic, 692
  - activemq::commands::BooleanExpression, 855
  - activemq::commands::BrokerError, 864
  - activemq::commands::BrokerId, 870
  - activemq::commands::BrokerInfo, 897

- activemq::commands::ConnectionControl, 1268
- activemq::commands::ConnectionError, 1297
- activemq::commands::ConnectionId, 1328
- activemq::commands::ConnectionInfo, 1356
- activemq::commands::ConsumerControl, 1402
- activemq::commands::ConsumerId, 1431
- activemq::commands::ConsumerInfo, 1461
- activemq::commands::ControlCommand, 1494
- activemq::commands::DataArrayResponse, 1529
- activemq::commands::DataResponse, 1584
- activemq::commands::DataStructure, 1660
- activemq::commands::DestinationInfo, 1728
- activemq::commands::DiscoveryEvent, 1759
- activemq::commands::ExceptionResponse, 1840
- activemq::commands::FlushCommand, 1938
- activemq::commands::IntegerResponse, 2092
- activemq::commands::JournalQueueAck, 2157
- activemq::commands::JournalTopicAck, 2184
- activemq::commands::JournalTrace, 2213
- activemq::commands::JournalTransaction, 2240
- activemq::commands::KeepAliveInfo, 2267
- activemq::commands::LastPartialCommand, 2301
- activemq::commands::LocalTransactionId, 2348
- activemq::commands::Message, 2520
- activemq::commands::MessageAck, 2560
- activemq::commands::MessageDispatch, 2595
- activemq::commands::MessageDispatchNotification, 2631
- activemq::commands::MessageId, 2664
- activemq::commands::MessagePull, 2740
- activemq::commands::NetworkBridgeFilter, 2794
- activemq::commands::PartialCommand, 2919
- activemq::commands::ProducerAck, 3037
- activemq::commands::ProducerId, 3068
- activemq::commands::ProducerInfo, 3097
- activemq::commands::RemoveInfo, 3195
- activemq::commands::RemoveSubscriptionInfo, 3223
- activemq::commands::ReplayCommand, 3252
- activemq::commands::Response, 3286
- activemq::commands::SessionId, 3380
- activemq::commands::SessionInfo, 3408
- activemq::commands::ShutdownInfo, 3471
- activemq::commands::SubscriptionInfo, 3672
- activemq::commands::TransactionId, 3820
- activemq::commands::TransactionInfo, 3848
- activemq::commands::WireFormatInfo, 3985
- activemq::commands::XATransactionId, 4032
- close
  - activemq::cmsutil::CachedConsumer, 1072
  - activemq::cmsutil::CachedProducer, 1076
  - activemq::cmsutil::PooledSession, 2957
  - activemq::commands::ActiveMQTempDestination, 577
  - activemq::commands::ConnectionControl, 1271
  - activemq::commands::ConsumerControl, 1405
  - activemq::core::ActiveMQConnection, 279
  - activemq::core::ActiveMQConsumer, 313
  - activemq::core::ActiveMQProducer, 472
  - activemq::core::ActiveMQQueueBrowser, 488
  - activemq::core::ActiveMQSession, 520
  - activemq::core::ActiveMQSessionExecutor, 534
  - activemq::core::MessageDispatchChannel, 2600
  - activemq::transport::correlator::ResponseCorrelator, 3292
  - activemq::transport::failover::FailoverTransport, 1876
  - activemq::transport::inactivity::InactivityMonitor, 2005
  - activemq::transport::IOTransport, 2147
  - activemq::transport::mock::MockTransport, 2770
  - activemq::transport::tcp::TcpTransport, 3753
  - activemq::transport::TransportFilter, 3893
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2902
- cms::Closeable, 1148
- cms::Connection, 1263

- cms::Session, 3368
- decaf::internal::io::StandardErrorOutputStream, 3580
- decaf::internal::io::StandardOutputStream, 3584
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2863
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2886
- decaf::internal::net::tcp::TcpSocket, 3742
- decaf::internal::net::tcp::TcpSocketInputStream, 3748
- decaf::internal::net::tcp::TcpSocketOutputStream, 3751
- decaf::io::BlockingByteArrayInputStream, 841
- decaf::io::BufferedInputStream, 937
- decaf::io::Closeable, 1149
- decaf::io::FilterInputStream, 1896
- decaf::io::FilterOutputStream, 1901
- decaf::io::InputStream, 2045
- decaf::io::InputStreamReader, 2054
- decaf::io::OutputStream, 2909
- decaf::io::OutputStreamWriter, 2916
- decaf::net::ServerSocket, 3357
- decaf::net::Socket, 3509
- decaf::net::SocketImpl, 3532
- decaf::util::logging::ConsoleHandler, 1399
- decaf::util::logging::StreamHandler, 3650
- decaf::util::zip::DeflaterOutputStream, 1718
- decaf::util::zip::InflaterInputStream, 2039
- CLOSE\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- closed
  - decaf::io::FilterInputStream, 1899
  - decaf::io::FilterOutputStream, 1903
- CloseTransportsTask
  - activemq::transport::failover::CloseTransportsTask, 1151
- cluster
  - activemq::commands::Message, 2532
- cms, 140
- cms/Config.h
  - CMS\_API, 4228
- cms::BytesMessage, 1056
  - ~BytesMessage, 1059
  - clone, 1059
  - getBodyBytes, 1059
  - getBodyLength, 1060
  - readBoolean, 1060
  - readByte, 1060
  - readBytes, 1061
  - readChar, 1062
  - readDouble, 1062
  - readFloat, 1062
  - readInt, 1063
  - readLong, 1063
  - readShort, 1063
  - readString, 1064
  - readUnsignedShort, 1064
  - reset, 1065
  - setBodyBytes, 1065
  - writeBoolean, 1065
  - writeByte, 1066
  - writeBytes, 1066
  - writeChar, 1067
  - writeDouble, 1067
  - writeFloat, 1067
  - writeInt, 1068
  - writeLong, 1068
  - writeShort, 1068
  - writeString, 1069
  - writeUnsignedShort, 1069
  - writeUTF, 1069
- cms::Closeable, 1148
  - ~Closeable, 1148
  - close, 1148
- cms::CMSException, 1160
  - ~CMSException, 1161
  - CMSException, 1161
  - getCause, 1161
  - getMessage, 1161
  - getStackTrace, 1161
  - getStackTraceString, 1162
  - printStackTrace, 1162
  - setMark, 1162
  - what, 1162
- cms::CMSProperties, 1165
  - ~CMSProperties, 1166
  - clear, 1166
  - clone, 1166
  - copy, 1166
  - getProperty, 1166
  - hasProperty, 1167
  - isEmpty, 1167
  - remove, 1167
  - setProperty, 1167
  - toArray, 1168
  - toString, 1168
- cms::CMSSecurityException, 1169
  - ~CMSSecurityException, 1169
  - CMSSecurityException, 1169
- cms::Connection, 1262
  - ~Connection, 1263

- close, 1263
- createSession, 1263, 1264
- getClientID, 1264
- getExceptionListener, 1264
- getMetaData, 1264
- setClientID, 1265
- setExceptionListener, 1265
- cms::ConnectionFactory, 1324
  - ~ConnectionFactory, 1325
  - createCMSConnectionFactory, 1325
  - createConnection, 1325, 1326
- cms::ConnectionMetaData, 1385
  - ~ConnectionMetaData, 1386
  - getCMSMajorVersion, 1386
  - getCMSMinorVersion, 1386
  - getCMSProviderName, 1386
  - getCMSVersion, 1386
  - getCMSXPropertyNames, 1387
  - getProviderMajorVersion, 1387
  - getProviderMinorVersion, 1387
  - getProviderVersion, 1388
- cms::DeliveryMode, 1721
  - ~DeliveryMode, 1722
  - DELIVERY\_MODE, 1721
  - NON\_PERSISTENT, 1721
  - PERSISTENT, 1721
- cms::Destination, 1723
  - ~Destination, 1724
  - clone, 1724
  - copy, 1724
  - DestinationType, 1723
  - getCMSProperties, 1724
  - getDestinationType, 1724
  - QUEUE, 1723
  - TEMPORARY\_QUEUE, 1723
  - TEMPORARY\_TOPIC, 1723
  - TOPIC, 1723
- cms::ExceptionListener, 1838
  - ~ExceptionListener, 1838
  - onException, 1838
- cms::IllegalStateException, 1997
  - ~IllegalStateException, 1997
  - IllegalStateException, 1997
- cms::InvalidClientIdException, 2130
  - ~InvalidClientIdException, 2130
  - InvalidClientIdException, 2130
- cms::InvalidDestinationException, 2131
  - ~InvalidDestinationException, 2131
  - InvalidDestinationException, 2131
- cms::InvalidSelectorException, 2138
  - ~InvalidSelectorException, 2138
  - InvalidSelectorException, 2138
- cms::MapMessage, 2472
  - ~MapMessage, 2474
  - getBoolean, 2474
  - getByte, 2474
  - getBytes, 2475
  - getChar, 2475
  - getDouble, 2475
  - getFloat, 2476
  - getInt, 2476
  - getLong, 2476
  - getMapNames, 2477
  - getShort, 2477
  - getString, 2477
  - itemExists, 2478
  - setBoolean, 2478
  - setByte, 2478
  - setBytes, 2479
  - setChar, 2479
  - setDouble, 2479
  - setFloat, 2480
  - setInt, 2480
  - setLong, 2480
  - setShort, 2481
  - setString, 2481
- cms::Message, 2534
  - ~Message, 2538
  - acknowledge, 2538
  - clearBody, 2538
  - clearProperties, 2539
  - clone, 2539
  - getBooleanProperty, 2540
  - getByteProperty, 2540
  - getCMSCorrelationID, 2540
  - getCMSDeliveryMode, 2541
  - getCMSDestination, 2541
  - getCMSExpiration, 2542
  - getCMSMessageID, 2542
  - getCMSPriority, 2543
  - getCMSRedelivered, 2543
  - getCMSReplyTo, 2544
  - getCMSTimestamp, 2544
  - getCMSType, 2545
  - getDoubleProperty, 2545
  - getFloatProperty, 2546
  - getIntProperty, 2546
  - getLongProperty, 2547
  - getPropertyNames, 2547
  - getShortProperty, 2548
  - getStringProperty, 2548
  - propertyExists, 2549
  - setBooleanProperty, 2549
  - setByteProperty, 2550
  - setCMSCorrelationID, 2550
  - setCMSDeliveryMode, 2551
  - setCMSDestination, 2551
  - setCMSExpiration, 2552

- setCMSMessageID, 2552
- setCMSPriority, 2552
- setCMSRedelivered, 2553
- setCMSReplyTo, 2553
- setCMSTimestamp, 2554
- setCMSType, 2554
- setDoubleProperty, 2555
- setFloatProperty, 2555
- setIntProperty, 2556
- setLongProperty, 2556
- setShortProperty, 2557
- setStringProperty, 2557
- cms::MessageConsumer, 2589
  - ~MessageConsumer, 2590
  - getMessageListener, 2590
  - getMessageSelector, 2590
  - receive, 2590, 2591
  - receiveNoWait, 2591
  - setMessageListener, 2591
- cms::MessageEnumeration, 2659
  - ~MessageEnumeration, 2659
  - hasMoreMessages, 2659
  - nextMessage, 2659
- cms::MessageEOFException, 2661
  - ~MessageEOFException, 2661
  - MessageEOFException, 2661
- cms::MessageFormatException, 2662
  - ~MessageFormatException, 2662
  - MessageFormatException, 2662
- cms::MessageListener, 2692
  - ~MessageListener, 2692
  - onMessage, 2692
- cms::MessageNotReadableException, 2723
  - ~MessageNotReadableException, 2723
  - MessageNotReadableException, 2723
- cms::MessageNotWriteableException, 2724
  - ~MessageNotWriteableException, 2724
  - MessageNotWriteableException, 2724
- cms::MessageProducer, 2725
  - ~MessageProducer, 2726
  - getDeliveryMode, 2726
  - getDisableMessageID, 2727
  - getDisableMessageTimeStamp, 2727
  - getPriority, 2727
  - getTimeToLive, 2727
  - send, 2728, 2729
  - setDeliveryMode, 2730
  - setDisableMessageID, 2730
  - setDisableMessageTimeStamp, 2730
  - setPriority, 2731
  - setTimeToLive, 2731
- cms::ObjectMessage, 2841
  - ~ObjectMessage, 2841
- cms::Queue, 3148
  - ~Queue, 3148
  - getQueueName, 3148
- cms::QueueBrowser, 3153
  - ~QueueBrowser, 3153
  - getEnumeration, 3153
  - getMessageSelector, 3154
  - getQueue, 3154
- cms::Session, 3365
  - ~Session, 3368
  - AcknowledgeMode, 3368
  - AUTO\_ACKNOWLEDGE, 3368
  - CLIENT\_ACKNOWLEDGE, 3368
  - close, 3368
  - commit, 3369
  - createBrowser, 3369
  - createBytesMessage, 3370
  - createConsumer, 3370, 3371
  - createDurableConsumer, 3371
  - createMapMessage, 3372
  - createMessage, 3372
  - createProducer, 3372
  - createQueue, 3373
  - createStreamMessage, 3373
  - createTemporaryQueue, 3373
  - createTemporaryTopic, 3374
  - createTextMessage, 3374
  - createTopic, 3374
  - DUPS\_OK\_ACKNOWLEDGE, 3368
  - getAcknowledgeMode, 3375
  - INDIVIDUAL\_ACKNOWLEDGE, 3368
  - isTransacted, 3375
  - recover, 3375
  - rollback, 3376
  - SESSION\_TRANSACTED, 3368
  - unsubscribe, 3376
- cms::Startable, 3586
  - ~Startable, 3586
  - start, 3586
- cms::Stoppable, 3648
  - ~Stoppable, 3648
  - stop, 3648
- cms::StreamMessage, 3652
  - ~StreamMessage, 3655
  - readBoolean, 3655
  - readByte, 3655
  - readBytes, 3655, 3656
  - readChar, 3657
  - readDouble, 3657
  - readFloat, 3657
  - readInt, 3658
  - readLong, 3658
  - readShort, 3659
  - readString, 3659
  - readUnsignedShort, 3659



- writeBoolean, 3660
- writeByte, 3660
- writeBytes, 3661
- writeChar, 3661
- writeDouble, 3662
- writeFloat, 3662
- writeInt, 3662
- writeLong, 3663
- writeShort, 3663
- writeString, 3663
- writeUnsignedShort, 3664
- cms::TemporaryQueue, 3759
  - ~TemporaryQueue, 3759
  - destroy, 3759
  - getQueueName, 3759
- cms::TemporaryTopic, 3761
  - ~TemporaryTopic, 3761
  - destroy, 3761
  - getTopicName, 3761
- cms::TextMessage, 3763
  - ~TextMessage, 3763
  - getText, 3763
  - setText, 3764
- cms::Topic, 3817
  - ~Topic, 3817
  - getTopicName, 3817
- cms::UnsupportedOperationException, 3919
  - ~UnsupportedOperationException, 3919
- UnsupportedOperationException, 3919
- CMS\_API
  - cms/Config.h, 4228
- CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1154
- CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1158
- CMSException
  - cms::CMSException, 1161
- CMSExceptionSupport.h
  - AMQ\_CATCH\_ALL\_THROW\_-CMSEXCEPTION, 4224
- CMSSecurityException
  - cms::CMSSecurityException, 1169
- CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1173
- Code
  - deflate.h, 4727
- code, 1183
  - bits, 1183
  - ct\_data\_s, 1527
  - op, 1183
  - val, 1183
- CODELENS
  - inflate.h, 4733
- CODES
  - inftrees.h, 4734
- codes
  - inflate\_state, 2025
- codetype
  - inftrees.h, 4734
- comm\_max
  - gz\_header\_s, 1975
- command
  - activemq::commands::ControlCommand, 1496
- commandId
  - activemq::commands::PartialCommand, 2921
- COMMENT
  - inflate.h, 4732
- comment
  - gz\_header\_s, 1975
- COMMENT\_STATE
  - deflate.h, 4727
- COMMIT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- commit
  - activemq::cmsutil::PooledSession, 2958
  - activemq::core::ActiveMQConsumer, 313
  - activemq::core::ActiveMQSession, 520
  - activemq::core::ActiveMQTransactionContext, 720
  - cms::Session, 3369
- compact
  - decaf::internal::nio::ByteBuffer, 1007
  - decaf::internal::nio::CharArrayBuffer, 1114
  - decaf::internal::nio::DoubleArrayBuffer, 1804
  - decaf::internal::nio::FloatArrayBuffer, 1920
  - decaf::internal::nio::IntArrayBuffer, 2061
  - decaf::internal::nio::LongArrayBuffer, 2439
  - decaf::internal::nio::ShortArrayBuffer, 3454
  - decaf::nio::ByteBuffer, 1040
  - decaf::nio::CharBuffer, 1125
  - decaf::nio::DoubleBuffer, 1813
  - decaf::nio::FloatBuffer, 1929
  - decaf::nio::IntBuffer, 2070
  - decaf::nio::LongBuffer, 2448
  - decaf::nio::ShortBuffer, 3463
- COMPARATOR
  - activemq::commands::BrokerId, 870
  - activemq::commands::ConnectionId, 1328
  - activemq::commands::ConsumerId, 1431
  - activemq::commands::LocalTransactionId, 2348
  - activemq::commands::MessageId, 2664

- activemq::commands::ProducerId, 3068
- activemq::commands::SessionId, 3380
- activemq::commands::TransactionId, 3819
- activemq::commands::XATransactionId, 4032
- comparator
  - decaf::util::PriorityQueue, 3031
- compare
  - activemq::util::IdGenerator, 1989
  - decaf::lang::ArrayPointerComparator, 737
  - decaf::lang::Double, 1790
  - decaf::lang::Float, 1906
  - decaf::lang::PointerComparator, 2954
  - decaf::util::Comparator, 1217
  - decaf::util::comparators::Less, 2328
- compareAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 739
  - decaf::util::concurrent::atomic::AtomicInteger, 743
  - decaf::util::concurrent::atomic::AtomicReference, 750
- compareTo
  - activemq::commands::BrokerId, 870
  - activemq::commands::ConnectionId, 1328
  - activemq::commands::ConsumerId, 1431
  - activemq::commands::LocalTransactionId, 2348
  - activemq::commands::MessageId, 2664
  - activemq::commands::ProducerId, 3068
  - activemq::commands::SessionId, 3380
  - activemq::commands::TransactionId, 3820
  - activemq::commands::XATransactionId, 4032
  - decaf::lang::Boolean, 851
  - decaf::lang::Byte, 962
  - decaf::lang::Character, 1102
  - decaf::lang::Comparable, 1214
  - decaf::lang::Double, 1791
  - decaf::lang::Float, 1907
  - decaf::lang::Integer, 2080
  - decaf::lang::Long, 2423
  - decaf::lang::Short, 3442
  - decaf::net::URI, 3925
  - decaf::nio::ByteBuffer, 1040
  - decaf::nio::CharBuffer, 1126
  - decaf::nio::DoubleBuffer, 1813
  - decaf::nio::FloatBuffer, 1929
  - decaf::nio::IntBuffer, 2070
  - decaf::nio::LongBuffer, 2448
  - decaf::nio::ShortBuffer, 3463
  - decaf::util::concurrent::TimeUnit, 3809
  - decaf::util::Date, 1667
  - decaf::util::logging::Level, 2334
  - decaf::util::UUID, 3971
- COMPOSITE\_SEPARATOR
  - activemq::commands::ActiveMQDestination, 331
- CompositeData
  - activemq::util::CompositeData, 1220
- CompositeTaskRunner
  - activemq::threads::CompositeTaskRunner, 1224
- compressed
  - activemq::commands::Message, 2532
- Concurrent.h
  - synchronized, 4893
  - WAIT\_INFINITE, 4893
- ConcurrentStlMap
  - decaf::util::concurrent::ConcurrentStlMap, 1237
- condition
  - decaf::util::concurrent::ConditionHandle, 1255
- ConditionHandle
  - decaf::util::concurrent::ConditionHandle, 1255
- CONFIG
  - decaf::util::logging::Level, 2335
- config
  - decaf::util::logging::Logger, 2389
- configure
  - activemq::core::PrefetchPolicy, 2977
  - activemq::core::RedeliveryPolicy, 3178
  - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2491
  - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2492
  - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2488
  - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2489
  - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2490
  - activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2487
- configureSocket
  - activemq::transport::tcp::SslTransport, 3575
  - activemq::transport::tcp::TcpTransport, 3753
- CONNECT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- connect
  - activemq::transport::tcp::TcpTransport, 3753

- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2863
- decaf::internal::net::tcp::TcpSocket, 3742
- decaf::net::Socket, 3509
- decaf::net::SocketImpl, 3532
- CONNECTED
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- connectedBrokers
  - activemq::commands::ConnectionControl, 1271
- ConnectException
  - decaf::net::ConnectException, 1259, 1260
- connection
  - activemq::commands::ActiveMQTempDestination, 579
  - activemq::commands::Message, 2532
- CONNECTION\_ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination, 331
- CONNECTION\_ALWAYS\_SYNC\_SEND
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_CLOSE\_TIMEOUT
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_DISPATCH\_ASYNC
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_PRODUCER\_WINDOW\_SIZE
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_SEND\_TIMEOUT
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_USE\_ASYNC\_SEND
  - activemq::core::ActiveMQConstants, 309
- CONNECTION\_USE\_COMPRESSION
  - activemq::core::ActiveMQConstants, 309
- ConnectionControl
  - activemq::commands::ConnectionControl, 1268
- ConnectionControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1281
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1293
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1273
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1277
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1285
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1289
- ConnectionError
  - activemq::commands::ConnectionError, 1297
- ConnectionErrorMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1313
  - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1301
  - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1305
  - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1309
  - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1317
  - activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1321
- ConnectionId
  - activemq::commands::ConnectionId, 1328
- ConnectionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1344
  - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1332
  - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1336
  - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1340
  - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1348
  - activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1352
- ConnectionInfo
  - activemq::commands::BrokerInfo, 903
  - activemq::commands::ConnectionError, 1299
  - activemq::commands::ConnectionInfo, 1360
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::DestinationInfo, 1731
  - activemq::commands::LocalTransactionId, 2350
  - activemq::commands::ProducerId, 3070
  - activemq::commands::RemoveSubscriptionInfo, 3226
  - activemq::commands::SessionId, 3382
  - activemq::commands::TransactionInfo, 3850
- ConnectionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1374
  - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1362
  - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1366

- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1431
- 1370
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1405
- 1378
- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1466
- 1382
- activemq::commands::MessageAck, 2564
- connectionInterruptProcessingComplete
- activemq::state::ConnectionStateTracker, 2598
- 1394
- activemq::commands::MessageDispatchNotification, 2634
- ConnectionState
- activemq::state::ConnectionState, 1390
- ConnectionStateTracker
- activemq::state::ConnectionStateTracker, 1394
- ConsoleHandler
- decaf::util::logging::ConsoleHandler, 1399
- const
- zconf.h, 4738
- ConstReferenceType
- decaf::lang::ArrayPointer, 732
- CONSUMER\_ADVISORY\_PREFIX
- activemq::commands::ActiveMQDestination, 331
- CONSUMER\_DISPATCHASYNC
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_EXCLUSIVE
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_NOLOCAL
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_PREFETCHSIZE
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_PRIORITY
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_RETROACTIVE
- activemq::core::ActiveMQConstants, 308
- CONSUMER\_SELECTOR
- activemq::core::ActiveMQConstants, 308
- ConsumerControl
- activemq::commands::ConsumerControl, 1402
- ConsumerControlMarshaller
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1419
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1407
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1411
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1415
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1423
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1427
- ConsumerId
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1448
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1436
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1440
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1444
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1452
- activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1456
- ConsumerInfo
- activemq::commands::ConsumerInfo, 1461
- ConsumerInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1481
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1469
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1473
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1477
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1485
- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1489
- ConsumerState
- activemq::state::ConsumerState, 1492
- ConsumerControlMarshaller
- decaf::util::AbstractCollection, 184
- decaf::util::concurrent::SynchronousQueue, 3718
- ConsumerControlMarshaller
- decaf::util::StlList, 3596
- decaf::util::StlSet, 3626
- containsAll
- decaf::util::AbstractCollection, 184
- decaf::util::Collection, 1189
- decaf::util::SynchronousQueue, 3719
- containsKey

- decaf::util::concurrent::ConcurrentStlMap, 1238
- decaf::util::Map, 2461
- decaf::util::StlMap, 3606
- containsValue
  - decaf::util::concurrent::ConcurrentStlMap, 1238
  - decaf::util::Map, 2462
  - decaf::util::StlMap, 3607
- content
  - activemq::commands::Message, 2532
- ControlCommand
  - activemq::commands::ControlCommand, 1494
- ControlCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1510
  - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1498
  - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1502
  - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1506
  - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1514
  - activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1518
- convert
  - activemq::util::PrimitiveValueConverter, 3010
  - decaf::util::concurrent::TimeUnit, 3810
- convertConsumerId
  - activemq::wireformat::stomp::StompHelper, 3639
- convertDestination
  - activemq::wireformat::stomp::StompHelper, 3639, 3640
- convertMessageId
  - activemq::wireformat::stomp::StompHelper, 3640
- convertProducerId
  - activemq::wireformat::stomp::StompHelper, 3640, 3641
- convertProperties
  - activemq::wireformat::stomp::StompHelper, 3641
- convertToCMSException
  - activemq::exceptions::ActiveMQException, 358
- convertTransactionId
  - activemq::wireformat::stomp::StompHelper, 3641, 3642
- COPY
  - gzguts.h, 4729
  - inflate.h, 4733
- copy
  - activemq::commands::ActiveMQQueue, 484
  - activemq::commands::ActiveMQTempQueue, 605
  - activemq::commands::ActiveMQTempTopic, 634
  - activemq::commands::ActiveMQTopic, 692
  - activemq::util::ActiveMQProperties, 479
  - activemq::wireformat::stomp::StompFrame, 3634
  - cms::CMSProperties, 1166
  - cms::Destination, 1724
  - decaf::util::AbstractCollection, 185
  - decaf::util::concurrent::ConcurrentStlMap, 1238, 1239
  - decaf::util::Map, 2463
  - decaf::util::Properties, 3128
  - decaf::util::StlList, 3396
  - decaf::util::StlMap, 3607
  - decaf::util::StlSet, 3627
- COPY
  - ControlCommandMarshaller, inflate.h, 4733
  - copyDataStructure
    - activemq::commands::ActiveMQBlobMessage, 205
    - activemq::commands::ActiveMQBytesMessage, 237
    - activemq::commands::ActiveMQDestination, 325
    - activemq::commands::ActiveMQMapMessage, 364
    - activemq::commands::ActiveMQMessage, 399
    - activemq::commands::ActiveMQObjectMessage, 444
    - activemq::commands::ActiveMQQueue, 484
    - activemq::commands::ActiveMQStreamMessage, 540
    - activemq::commands::ActiveMQTempDestination, 577
    - activemq::commands::ActiveMQTempQueue, 605
    - activemq::commands::ActiveMQTempTopic, 634
    - activemq::commands::ActiveMQTextMessage, 664
    - activemq::commands::ActiveMQTopic, 692
    - activemq::commands::BaseCommand, 759
    - activemq::commands::BaseDataStructure, 831

- activemq::commands::BooleanExpression, 855
- activemq::commands::BrokerError, 864
- activemq::commands::BrokerId, 870
- activemq::commands::BrokerInfo, 898
- activemq::commands::ConnectionControl, 1268
- activemq::commands::ConnectionError, 1297
- activemq::commands::ConnectionId, 1328
- activemq::commands::ConnectionInfo, 1356
- activemq::commands::ConsumerControl, 1402
- activemq::commands::ConsumerId, 1431
- activemq::commands::ConsumerInfo, 1461
- activemq::commands::ControlCommand, 1494
- activemq::commands::DataArrayResponse, 1529
- activemq::commands::DataResponse, 1584
- activemq::commands::DataStructure, 1661
- activemq::commands::DestinationInfo, 1728
- activemq::commands::DiscoveryEvent, 1759
- activemq::commands::ExceptionResponse, 1840
- activemq::commands::FlushCommand, 1938
- activemq::commands::IntegerResponse, 2092
- activemq::commands::JournalQueueAck, 2157
- activemq::commands::JournalTopicAck, 2184
- activemq::commands::JournalTrace, 2213
- activemq::commands::JournalTransaction, 2240
- activemq::commands::KeepAliveInfo, 2267
- activemq::commands::LastPartialCommand, 2302
- activemq::commands::LocalTransactionId, 2348
- activemq::commands::Message, 2521
- activemq::commands::MessageAck, 2560
- activemq::commands::MessageDispatch, 2595
- activemq::commands::MessageDispatchNotification, 2631
- activemq::commands::MessageId, 2665
- activemq::commands::MessagePull, 2740
- activemq::commands::NetworkBridgeFilter, 2794
- activemq::commands::PartialCommand, 2919
- activemq::commands::ProducerAck, 3037
- activemq::commands::ProducerId, 3068
- activemq::commands::ProducerInfo, 3097
- activemq::commands::RemoveInfo, 3195
- activemq::commands::RemoveSubscriptionInfo, 3223
- activemq::commands::ReplayCommand, 3252
- activemq::commands::Response, 3286
- activemq::commands::SessionId, 3380
- activemq::commands::SessionInfo, 3408
- activemq::commands::ShutdownInfo, 3471
- activemq::commands::SubscriptionInfo, 3672
- activemq::commands::TransactionId, 3820
- activemq::commands::TransactionInfo, 3848
- activemq::commands::WireFormatInfo, 3985
- activemq::commands::XATransactionId, 4032
- correlationId
  - activemq::commands::Message, 2532
  - activemq::commands::MessagePull, 2743
  - activemq::commands::Response, 3288
- countDown
  - decaf::util::concurrent::CountDownLatch, 1523
- CountDownLatch
  - decaf::util::concurrent::CountDownLatch, 1521
- CounterType
  - decaf::lang::ArrayPointer, 732
  - decaf::lang::Pointer, 2948
- countTokens
  - decaf::util::StringTokenizer, 3669
- CRC32
  - decaf::util::zip::CRC32, 1525
- crc32.h
  - crc\_table, 4724
- crc\_table
  - crc32.h, 4724
- create
  - activemq::transport::failover::FailoverTransportFactory, 1886
  - activemq::transport::mock::MockTransportFactory, 2780
  - activemq::transport::tcp::TcpTransportFactory, 3757
  - activemq::transport::TransportFactory, 3889

- activemq::util::CMSExceptionSupport, 1163
- decaf::internal::net::tcp::TcpSocket, 3742
- decaf::internal::util::concurrent::ConditionImpl, 1257
- decaf::internal::util::concurrent::MutexImpl, 2788
- decaf::net::SocketImpl, 3532
- decaf::net::URI, 3925
- createBrowser
  - activemq::cmsutil::PooledSession, 2958
  - activemq::core::ActiveMQSession, 520
  - cms::Session, 3369
- createByteBuffer
  - decaf::internal::nio::BufferFactory, 944, 945
- createBytesMessage
  - activemq::cmsutil::PooledSession, 2959
  - activemq::core::ActiveMQSession, 521
  - cms::Session, 3370
- createCachedConsumer
  - activemq::cmsutil::PooledSession, 2959
- createCachedProducer
  - activemq::cmsutil::PooledSession, 2960
- createCharBuffer
  - decaf::internal::nio::BufferFactory, 945, 946
- createCMSConnectionFactory
  - cms::ConnectionFactory, 1325
- createComposite
  - activemq::transport::failover::FailoverTransportFactory, 1886
  - activemq::transport::mock::MockTransportFactory, 2780
  - activemq::transport::tcp::TcpTransportFactory, 3757
  - activemq::transport::TransportFactory, 3890
- createConnection
  - activemq::cmsutil::CmsAccessor, 1154
  - activemq::core::ActiveMQConnectionFactory, 295, 296
  - cms::ConnectionFactory, 1325, 1326
- createConsumer
  - activemq::cmsutil::PooledSession, 2960, 2961
  - activemq::core::ActiveMQSession, 521, 522
  - cms::Session, 3370, 3371
- createDestination
  - activemq::commands::ActiveMQDestination, 325
- createDoubleBuffer
  - decaf::internal::nio::BufferFactory, 946, 947
- createDurableConsumer
  - activemq::cmsutil::PooledSession, 2961
  - activemq::core::ActiveMQSession, 522
- cms::Session, 3371
- createFloatBuffer
  - decaf::internal::nio::BufferFactory, 948
- createIntBuffer
  - decaf::internal::nio::BufferFactory, 949, 950
- createLongBuffer
  - decaf::internal::nio::BufferFactory, 950, 951
- createMapMessage
  - activemq::cmsutil::PooledSession, 2962
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3372
- createMessage
  - activemq::cmsutil::MessageCreator, 2593
  - activemq::cmsutil::PooledSession, 2962
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3372
- createMessageEOFException
  - activemq::util::CMSExceptionSupport, 1163
- createMessageFormatException
  - activemq::util::CMSExceptionSupport, 1163
- createNegotiator
  - activemq::wireformat::openwire::OpenWireFormat, 2890
  - activemq::wireformat::stomp::StompWireFormat, 3644
  - activemq::wireformat::WireFormat, 3977
- createObject
  - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1611
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 214
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesL, 254
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 379
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 406
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 451
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 496
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 557
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempO, 614
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempT, 647
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextM, 676
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicL, 704

activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2689
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2761
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2817
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2943
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	3061
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	3093
activemq::wireformat::openwire::marshal::v1::ConsumerGroupViewMarshaller	3110
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	3211
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3228
activemq::wireformat::openwire::marshal::v1::ContainerCommandMarshaller	3259
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3316
activemq::wireformat::openwire::marshal::v1::DataResponseViewMarshaller	3404
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3420
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	3482
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3681
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3856
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	4009
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	4049
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	222
activemq::wireformat::openwire::marshal::v1::JournalTopicInfoMarshaller	270
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	391
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	418
activemq::wireformat::openwire::marshal::v1::LastReceivedCommandMarshaller	463
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	508
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	569
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	626
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	655
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFormatMarshaller	
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTemp5Marshaller	



activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	688	activemq::wireformat::openwire::marshal::v2::MessageDispatcher	2607
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMessageMarshaller	716	activemq::wireformat::openwire::marshal::v2::MessageDispatcher	2640
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	893	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	925	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2745
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1293	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilter	2797
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	1301	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2927
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1332	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	3041
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1362	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	3073
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	1407	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	3106
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1436	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	3199
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1469	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	3236
activemq::wireformat::openwire::marshal::v2::ContactCommandMarshaller	1498	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	3263
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1532	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	3301
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1595	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	3384
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1733	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	3428
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	1767	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	3478
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1847	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3697
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1945	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3872
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	2099	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	4001
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	2164	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	4041
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	2193	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	210
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	2216	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	250
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	2247	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	375
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	2274	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	402
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	2313	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller	447
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	2356	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	492
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2570	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller	553

activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2360	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	2360
610		activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2574
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2574	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2611
639		activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2611
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2611	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2644
668		activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2644
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMessageMarshaller	2644	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2681
696		activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2681
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2681	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2753
873		activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2753
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2753	activemq::wireformat::openwire::marshal::v3::NetworkBridgeMarshaller	2809
905		activemq::wireformat::openwire::marshal::v3::NetworkBridgeMarshaller	2809
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2809	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2935
1273		activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2935
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2935	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	3049
1305		activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	3049
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	3049	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	3081
1336		activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	3081
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	3081	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	3118
1366		activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	3118
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWithMarshaller	3118	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	3207
1411		activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	3207
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	3207	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	3232
1440		activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	3232
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	3232	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	3267
1473		activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	3267
activemq::wireformat::openwire::marshal::v3::ContactCommandMarshaller	3267	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3311
1502		activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3311
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	3311	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3400
1536		activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3400
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3400	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3424
1599		activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3424
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	3424	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3490
1737		activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3490
activemq::wireformat::openwire::marshal::v3::DiscardInfoMarshaller	3490	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3677
1771		activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3677
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	3677	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3860
1851		activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3860
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3860	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	4013
1949		activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	4013
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	4013	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	4053
2103		activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	4053
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	4053	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller	218
2172		activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller	218
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	218	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller	258
2197		activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller	258
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	258	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller	383
2220		activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller	383
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	383	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	410
2251		activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	410
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	410	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller	455
2278		activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller	455
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	455		
2309			

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	500	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	2282
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	561	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	2321
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller	618	activemq::wireformat::openwire::marshal::v4::LocalTransactionInfoMarshaller	2368
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueTopicMarshaller	643	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2578
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueTopicMarshaller	672	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2619
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueTopicMarshaller	700	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2648
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	877	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2673
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	909	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2757
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1277	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFormatMarshaller	2813
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1309	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2939
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1340	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	3045
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1370	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	3077
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1415	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	3102
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1444	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	3219
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1477	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller	3248
activemq::wireformat::openwire::marshal::v4::ContainerCommandMarshaller	1506	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	3255
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	1540	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	3296
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1603	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	3388
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1741	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	3432
activemq::wireformat::openwire::marshal::v4::DiscardInfoMarshaller	1775	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	3494
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1859	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	3689
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1953	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3868
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	2107	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	4005
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2176	activemq::wireformat::openwire::marshal::v4::XATransactionInfoMarshaller	4045
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2205	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMarshaller	226
activemq::wireformat::openwire::marshal::v4::JournalQueueMarshaller	2228	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMarshaller	262
activemq::wireformat::openwire::marshal::v4::JournalQueueMarshaller	2259	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMarshaller	387

activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller;openwire::marshal::v5::JournalTraceMa	2236
414	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller;marshal::v5::JournalTransact	2255
459	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller;openwire::marshal::v5::KeepAliveInfoM	2286
504	
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller;marshal::v5::LastPartialComm	2317
565	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller;openwire::marshal::v5::LocalTransaction	2364
622	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller;openwire::marshal::v5::MessageAckMar	2586
651	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller;openwire::marshal::v5::MessageDispatc	2615
680	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller;openwire::marshal::v5::MessageDispatc	2656
708	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller;openwire::marshal::v5::MessageIdMarsh	2677
885	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller;openwire::marshal::v5::MessagePullMar	2749
917	
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller;openwire::marshal::v5::NetworkBridgeF	2805
1285	
activemq::wireformat::openwire::marshal::v5::ConnectionFromMarshaller;openwire::marshal::v5::PartialCommand	2931
1317	
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller;openwire::marshal::v5::ProducerAckMar	3053
1348	
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller;openwire::marshal::v5::ProducerIdMar	3085
1378	
activemq::wireformat::openwire::marshal::v5::ConsumerGroupMarshaller;openwire::marshal::v5::ProducerInfoMa	3114
1423	
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller;openwire::marshal::v5::RemoveInfoMar	3215
1452	
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;openwire::marshal::v5::RemoveSubscrip	3244
1485	
activemq::wireformat::openwire::marshal::v5::ContextCommandMarshaller;openwire::marshal::v5::ReplayCommand	3275
1514	
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller;openwire::marshal::v5::ResponseMarsha	3306
1548	
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller;openwire::marshal::v5::SessionIdMarsha	3396
1587	
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller;openwire::marshal::v5::SessionInfoMar	3416
1753	
activemq::wireformat::openwire::marshal::v5::DiscoveryInfoMarshaller;openwire::marshal::v5::ShutdownInfoMa	3486
1783	
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller;openwire::marshal::v5::SubscriptionInfo	3685
1855	
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller;openwire::marshal::v5::TransactionInfoM	3852
1961	
activemq::wireformat::openwire::marshal::v5::IntegrationResponseMarshaller;openwire::marshal::v5::WireFormatInfo	3993
2115	
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller;openwire::marshal::v5::XATransactionI	4057
2168	
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller;openwire::marshal::v6::ActiveMQBlobM	230
2189	

activemq::wireformat::openwire::marshal::v6::ActiveMQByteMessageMarshaller	266	activemq::wireformat::openwire::marshal::v6::ActiveMQByteMessageMarshaller	2160	activemq::wireformat::openwire::marshal::v6::JournalQueueAck	
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	395	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	2201	activemq::wireformat::openwire::marshal::v6::JournalTopicAck	
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	422	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	2224	activemq::wireformat::openwire::marshal::v6::JournalTraceMa	
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	467	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	2243	activemq::wireformat::openwire::marshal::v6::JournalTransact	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	512	activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	2270	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoM	
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	573	activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	2305	activemq::wireformat::openwire::marshal::v6::LastPartialComm	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	630	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	2352	activemq::wireformat::openwire::marshal::v6::LocalTransaction	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	659	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	2566	activemq::wireformat::openwire::marshal::v6::MessageAckMar	
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	684	activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	2627	activemq::wireformat::openwire::marshal::v6::MessageDispatc	
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	712	activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	2636	activemq::wireformat::openwire::marshal::v6::MessageDispatc	
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	889	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	2685	activemq::wireformat::openwire::marshal::v6::MessageIdMarsh	
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	921	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	2765	activemq::wireformat::openwire::marshal::v6::MessagePullMar	
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	1289	activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	2801	activemq::wireformat::openwire::marshal::v6::NetworkBridgeF	
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	1321	activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	2923	activemq::wireformat::openwire::marshal::v6::PartialCommand	
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1352	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	3057	activemq::wireformat::openwire::marshal::v6::ProducerAckMar	
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	1382	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	3089	activemq::wireformat::openwire::marshal::v6::ProducerIdMars	
activemq::wireformat::openwire::marshal::v6::ConsumerGroupMarshaller	1427	activemq::wireformat::openwire::marshal::v6::ConsumerGroupMarshaller	3122	activemq::wireformat::openwire::marshal::v6::ProducerInfoMa	
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1456	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	3203	activemq::wireformat::openwire::marshal::v6::RemoveInfoMars	
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	1489	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	3240	activemq::wireformat::openwire::marshal::v6::RemoveSubscrip	
activemq::wireformat::openwire::marshal::v6::ContentCommandMarshaller	1518	activemq::wireformat::openwire::marshal::v6::ContentCommandMarshaller	3271	activemq::wireformat::openwire::marshal::v6::ReplayCommand	
activemq::wireformat::openwire::marshal::v6::DataActiveResponseMarshaller	1552	activemq::wireformat::openwire::marshal::v6::DataActiveResponseMarshaller	3321	activemq::wireformat::openwire::marshal::v6::ResponseMarsha	
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	1591	activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	3392	activemq::wireformat::openwire::marshal::v6::SessionIdMarsha	
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	1749	activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	3412	activemq::wireformat::openwire::marshal::v6::SessionInfoMars	
activemq::wireformat::openwire::marshal::v6::DiscardEventMarshaller	1763	activemq::wireformat::openwire::marshal::v6::DiscardEventMarshaller	3474	activemq::wireformat::openwire::marshal::v6::ShutdownInfoMa	
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1843	activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	3693	activemq::wireformat::openwire::marshal::v6::SubscriptionInfo	
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	1941	activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	3864	activemq::wireformat::openwire::marshal::v6::TransactionInfo	
activemq::wireformat::openwire::marshal::v6::IntegrationResponseMarshaller	2095	activemq::wireformat::openwire::marshal::v6::IntegrationResponseMarshaller	3997	activemq::wireformat::openwire::marshal::v6::WireFormatInfo	

- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4037
- createProducer
  - activemq::cmsutil::PooledSession, 2962
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3372
- createQueryString
  - activemq::util::URISupport, 3945
- createQueue
  - activemq::cmsutil::PooledSession, 2963
  - activemq::core::ActiveMQSession, 524
  - cms::Session, 3373
- createRemoveCommand
  - activemq::commands::ConnectionInfo, 1357
  - activemq::commands::ConsumerInfo, 1461
  - activemq::commands::ProducerInfo, 3097
  - activemq::commands::SessionInfo, 3408
- createServerSocket
  - decaf::internal::net::DefaultServerSocketFactory, 1683, 1684
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1694, 1695
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2855, 2856
  - decaf::net::ServerSocketFactory, 3362, 3363
- createSession
  - activemq::cmsutil::CmsAccessor, 1155
  - activemq::core::ActiveMQConnection, 279
  - cms::Connection, 1263, 1264
- createShortBuffer
  - decaf::internal::nio::BufferFactory, 951, 952
- createSocket
  - activemq::transport::tcp::SslTransport, 3575
  - activemq::transport::tcp::TcpTransport, 3754
  - decaf::internal::net::DefaultSocketFactory, 1688–1690
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1699–1702
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2877–2880
  - decaf::net::SocketFactory, 3525–3527
  - decaf::net::ssl::SSLSocketFactory, 3572
- createSocketImpl
  - decaf::net::SocketImplFactory, 3537
- createStreamMessage
  - activemq::cmsutil::PooledSession, 2963
  - activemq::core::ActiveMQSession, 524
  - cms::Session, 3373
- createTemporaryName
  - activemq::commands::ActiveMQDestination, Dad 325
- createTemporaryQueue
  - activemq::cmsutil::PooledSession, 2963
  - activemq::core::ActiveMQSession, 524
  - cms::Session, 3373
- createTemporaryTopic
  - activemq::cmsutil::PooledSession, 2963
  - activemq::core::ActiveMQSession, 525
  - cms::Session, 3374
- createTextMessage
  - activemq::cmsutil::PooledSession, 2964
  - activemq::core::ActiveMQSession, 525
  - cms::Session, 3374
- createTopic
  - activemq::cmsutil::PooledSession, 2964
  - activemq::core::ActiveMQSession, 525
  - cms::Session, 3374
- createWireFormat
  - activemq::transport::AbstractTransportFactory, 201
  - activemq::wireformat::openwire::OpenWireFormatFactory, 2899
  - activemq::wireformat::stomp::StompWireFormatFactory, 3647
  - activemq::wireformat::WireFormatFactory, 3980
- criticalSection
  - decaf::util::concurrent::ConditionHandle, 1255
- ct\_data
  - deflate.h, 4727
- ct\_data\_s, 1527
- code, 1527
- dad, 1527
- dl, 1527
- fc, 1527
- freq, 1527
- len, 1527
- CUNSUMER\_MAXPENDINGMSGLIMIT
  - activemq::core::ActiveMQConstants, 308
- currentThread
  - decaf::lang::Thread, 3769
- currentTimeMillis
  - decaf::lang::System, 3730
- d\_buf
  - internal\_state, 2120
- d\_code
  - deflate.h, 4727
- D\_CODES
  - deflate.h, 4727
- d\_desc
  - internal\_state, 2120
- deflate.h, 4727

- dad
  - ct\_data\_s, 1527
- data
  - activemq::commands::DataArrayResponse, 1530
  - activemq::commands::DataResponse, 1585
  - activemq::commands::PartialCommand, 2921
- data\_type
  - z\_stream\_s, 4062
- DataArrayResponse
  - activemq::commands::DataArrayResponse, 1529
- DataArrayResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1544
  - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1532
  - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1536
  - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1540
  - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1548
  - activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1552
- DataFormatException
  - decaf::util::zip::DataFormatException, 1555, 1556
- DataInputStream
  - decaf::io::DataInputStream, 1567
- DataOutputStream
  - decaf::io::DataOutputStream, 1580
- DataResponse
  - activemq::commands::DataResponse, 1584
- DataResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1607
  - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1595
  - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1599
  - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1603
  - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1587
  - activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1591
- dataStructure
  - activemq::commands::Message, 2532
- Date
  - decaf::util::Date, 1666
- DAYS
  - decaf::util::concurrent::TimeUnit, 3815
- DEBUG
  - decaf::util::logging::Level, 2335
- Debug
  - decaf::util::logging, 176
- debug
  - decaf::util::logging::Logger, 2390
  - decaf::util::logging::SimpleLogger, 3502
- decaf, 143
- decaf/lang/exceptions/ExceptionDefines.h
  - DECAF\_CATCH\_EXCEPTION\_-CONVERT, 4173
  - DECAF\_CATCH\_NOTHROW, 4173
  - DECAF\_CATCH\_RETHROW, 4174
  - DECAF\_CATCHALL\_NOTHROW, 4174
  - DECAF\_CATCHALL\_THROW, 4174
- decaf/util/Config.h
  - DECAF\_USE\_ARRAYBOUNDED, 4229
  - DECAF\_UNUSED, 4229
- DecafRuntime
  - decaf::internal::AprPool, 728
  - AprPool, 728
  - decaf::internal::DecafRuntime, 1670
  - ~DecafRuntime, 1670
  - DecafRuntime, 1670
  - getGlobalPool, 1670
- decaf::internal::io, 145
- decaf::internal::io::StandardErrorOutputStream, 3579
  - ~StandardErrorOutputStream, 3580
  - close, 3580
  - doWriteArrayBounded, 3580
  - doWriteByte, 3580
  - flush, 3580
  - StandardErrorOutputStream, 3580
- decaf::internal::io::StandardInputStream, 3582
  - ~StandardInputStream, 3582
  - doReadByte, 3582
  - StandardInputStream, 3582
- decaf::internal::io::StandardOutputStream, 3584
  - ~StandardOutputStream, 3584
  - doWriteArrayBounded, 3585
  - doWriteByte, 3585
  - flush, 3585
  - StandardOutputStream, 3584
- decaf::internal::net, 146
- decaf::internal::net::DefaultServerSocketFactory, 1682

- ~DefaultServerSocketFactory, 1683
- createServerSocket, 1683, 1684
- DefaultServerSocketFactory, 1683
- decaf::internal::net::DefaultSocketFactory, 1686
  - ~DefaultSocketFactory, 1688
  - createSocket, 1688–1690
  - DefaultSocketFactory, 1688
- decaf::internal::net::Network, 2790
  - ~Network, 2791
  - addAsResource, 2791
  - addNetworkResource, 2791
  - getNetworkRuntime, 2791
  - getRuntimeLock, 2791
  - initializeNetworking, 2791
  - Network, 2791
  - shutdownNetworking, 2791
- decaf::internal::net::SocketFileDescriptor, 3528
  - ~SocketFileDescriptor, 3528
  - getValue, 3528
  - SocketFileDescriptor, 3528
- decaf::internal::net::ssl, 147
- decaf::internal::net::ssl::DefaultSSLContext, 1691
  - ~DefaultSSLContext, 1691
  - DefaultSSLContext, 1691
  - getContext, 1691
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1692
  - ~DefaultSSLServerSocketFactory, 1694
  - createServerSocket, 1694, 1695
  - DefaultSSLServerSocketFactory, 1694
  - getDefaultCipherSuites, 1695
  - getSupportedCipherSuites, 1696
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1697
  - ~DefaultSSLSocketFactory, 1699
  - createSocket, 1699–1702
  - DefaultSSLSocketFactory, 1699
  - getDefaultCipherSuites, 1702
  - getSupportedCipherSuites, 1702
- decaf::internal::net::ssl::openssl, 148
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2842
  - ~OpenSSLContextSpi, 2843
  - OpenSSLContextSpi, 2843
  - OpenSSLContextSpi, 2844
  - OpenSSLContextSpi, 2844
  - providerGetServerSocketFactory, 2843
  - providerGetSocketFactory, 2843
  - providerInit, 2844
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2845
  - ~OpenSSLParameters, 2845
  - clone, 2845
  - setEnabledCipherSuites, 2845
  - setEnabledProtocols, 2846
  - getNeedClientAuth, 2846
  - getSupportedCipherSuites, 2846
  - getSupportedProtocols, 2846
  - getUseClientMode, 2846
  - getWantClientAuth, 2846
  - setEnabledCipherSuites, 2846
  - setEnabledProtocols, 2846
  - setNeedClientAuth, 2846
  - setUseClientMode, 2846
  - setWantClientAuth, 2846
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2847
  - ~OpenSSLServerSocket, 2849
  - accept, 2849
  - setEnabledCipherSuites, 2849
  - setEnabledProtocols, 2849
  - getNeedClientAuth, 2850
  - getSupportedCipherSuites, 2850
  - getSupportedProtocols, 2850
  - getWantClientAuth, 2850
  - OpenSSLServerSocket, 2849
  - setEnabledCipherSuites, 2851
  - setEnabledProtocols, 2851
  - setNeedClientAuth, 2851
  - setWantClientAuth, 2852
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2853
  - ~OpenSSLServerSocketFactory, 2855
  - createServerSocket, 2855, 2856
  - getDefaultCipherSuites, 2856
  - getSupportedCipherSuites, 2857
  - OpenSSLServerSocketFactory, 2855
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2858
  - ~OpenSSLSocket, 2862
  - available, 2862
  - close, 2863
  - connect, 2863
  - setEnabledCipherSuites, 2863
  - setEnabledProtocols, 2864
  - getInputStream, 2864
  - getNeedClientAuth, 2864
  - getOutputStream, 2864
  - getSupportedCipherSuites, 2865
  - getSupportedProtocols, 2865
  - getUseClientMode, 2865
  - getWantClientAuth, 2866
  - OpenSSLSocket, 2862
  - read, 2866
  - sendUrgentData, 2866
  - setEnabledCipherSuites, 2867
  - setEnabledProtocols, 2867



- setNeedClientAuth, 2867
  - setOOBInline, 2868
  - setUseClientMode, 2868
  - setWantClientAuth, 2868
  - shutdownInput, 2869
  - shutdownOutput, 2869
  - startHandshake, 2869
  - write, 2869
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 3747
  - 2871
  - ~OpenSSLSocketException, 2873
  - clone, 2873
  - getErrorString, 2874
  - OpenSSLSocketException, 2872, 2873
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2875
  - ~OpenSSLSocketFactory, 2877
  - createSocket, 2877–2880
  - getDefaultCipherSuites, 2880
  - getSupportedCipherSuites, 2880
  - OpenSSLSocketFactory, 2877
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2882
  - ~OpenSSLSocketInputStream, 2883
  - available, 2883
  - close, 2883
  - doReadArrayBounded, 2883
  - doReadByte, 2883
  - OpenSSLSocketInputStream, 2883
  - skip, 2884
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2885
  - ~OpenSSLSocketOutputStream, 2886
  - close, 2886
  - doWriteArrayBounded, 2886
  - doWriteByte, 2886
  - OpenSSLSocketOutputStream, 2886
- decaf::internal::net::tcp, 149
- decaf::internal::net::tcp::TcpSocket, 3738
  - ~TcpSocket, 3741
  - accept, 3741
  - available, 3741
  - bind, 3741
  - checkResult, 3741
  - close, 3742
  - connect, 3742
  - create, 3742
  - getInputStream, 3742
  - getLocalAddress, 3743
  - getOption, 3743
  - getOutputStream, 3743
  - getSocketHandle, 3744
  - isClosed, 3744
  - isConnected, 3744
  - listen, 3744
  - read, 3744
  - setOption, 3745
  - shutdownInput, 3745
  - shutdownOutput, 3745
  - TcpSocket, 3741
  - write, 3746
- decaf::internal::net::tcp::TcpSocketInputStream, 3748
  - ~TcpSocketInputStream, 3748
  - available, 3748
  - close, 3748
  - doReadArrayBounded, 3748
  - doReadByte, 3749
  - skip, 3749
  - TcpSocketInputStream, 3748
- decaf::internal::net::tcp::TcpSocketOutputStream, 3750
  - ~TcpSocketOutputStream, 3750
  - close, 3751
  - doWriteArrayBounded, 3751
  - doWriteByte, 3751
  - TcpSocketOutputStream, 3750
- decaf::internal::net::URIEncoderDecoder, 3932
  - ~URIEncoderDecoder, 3932
  - decode, 3932
  - encodeOthers, 3933
  - quoteIllegal, 3933
  - URIEncoderDecoder, 3932
  - validate, 3933
- decaf::internal::net::URIHelper, 3935
  - ~URIHelper, 3936
  - isValidDomainName, 3937
  - isValidHexChar, 3937
  - isValidHost, 3937
  - isValidIP4Word, 3937
  - isValidIP6Address, 3938
  - isValidIPv4Address, 3938
  - parseAuthority, 3938
  - parseURI, 3939
  - URIHelper, 3936
  - validateAuthority, 3939
  - validateFragment, 3939
  - validatePath, 3939
  - validateQuery, 3940
  - validateScheme, 3940
  - validateSsp, 3940
  - validateUserinfo, 3941
- decaf::internal::net::URIType, 3952
  - ~URIType, 3954
  - getAuthority, 3954
  - getFragment, 3954
  - getHost, 3954

- getPath, 3954
- getPort, 3954
- getQuery, 3955
- getScheme, 3955
- getSchemeSpecificPart, 3955
- getSource, 3955
- getUserInfo, 3955
- isAbsolute, 3955
- isOpaque, 3956
- isServerAuthority, 3956
- isValid, 3956
- setAbsolute, 3956
- setAuthority, 3956
- setFragment, 3956
- setHost, 3957
- setOpaque, 3957
- setPath, 3957
- setPort, 3957
- setQuery, 3957
- setScheme, 3957
- setSchemeSpecificPart, 3958
- setServerAuthority, 3958
- setSource, 3958
- setUserInfo, 3958
- setValid, 3958
- URIType, 3954
- decaf::internal::nio, 150
- decaf::internal::nio::BufferFactory, 942
  - ~BufferFactory, 944
  - createByteBuffer, 944, 945
  - createCharBuffer, 945, 946
  - createDoubleBuffer, 946, 947
  - createFloatBuffer, 948
  - createIntBuffer, 949, 950
  - createLongBuffer, 950, 951
  - createShortBuffer, 951, 952
- decaf::internal::nio::ByteBuffer, 991
  - ~ByteBuffer, 1003
  - array, 1003
  - arrayOffset, 1004
  - asCharBuffer, 1004
  - asDoubleBuffer, 1004
  - asFloatBuffer, 1005
  - asIntBuffer, 1005
  - asLongBuffer, 1005
  - asReadOnlyBuffer, 1006
  - asShortBuffer, 1006
  - ByteBuffer, 1002, 1003
  - compact, 1007
  - duplicate, 1007
  - get, 1007, 1008
  - getChar, 1008
  - getDouble, 1009
  - getFloat, 1009, 1010
  - getInt, 1010
  - getLong, 1011
  - getShort, 1011, 1012
  - hasArray, 1012
  - isReadOnly, 1012
  - put, 1013
  - putChar, 1013, 1014
  - putDouble, 1014, 1015
  - putFloat, 1015, 1016
  - putInt, 1016
  - putLong, 1017
  - putShort, 1018
  - setReadOnly, 1019
  - slice, 1019
- decaf::internal::nio::CharArrayBuffer, 1108
  - ~CharArrayBuffer, 1113
  - \_array, 1118
  - array, 1113
  - arrayOffset, 1113
  - asReadOnlyBuffer, 1114
  - CharArrayBuffer, 1111, 1112
  - compact, 1114
  - duplicate, 1114
  - get, 1115
  - hasArray, 1115
  - isReadOnly, 1116
  - length, 1118
  - offset, 1118
  - put, 1116
  - readOnly, 1118
  - setReadOnly, 1117
  - slice, 1117
  - subSequence, 1117
- decaf::internal::nio::DoubleArrayBuffer, 1799
  - ~DoubleArrayBuffer, 1803
  - array, 1803
  - arrayOffset, 1804
  - asReadOnlyBuffer, 1804
  - compact, 1804
  - DoubleArrayBuffer, 1802, 1803
  - duplicate, 1805
  - get, 1805, 1806
  - hasArray, 1806
  - isReadOnly, 1806
  - put, 1806, 1807
  - setReadOnly, 1807
  - slice, 1807
- decaf::internal::nio::FloatArrayBuffer, 1915
  - ~FloatArrayBuffer, 1919
  - array, 1919
  - arrayOffset, 1920
  - asReadOnlyBuffer, 1920
  - compact, 1920
  - duplicate, 1921

- FloatArrayBuffer, 1918, 1919
- get, 1921, 1922
- hasArray, 1922
- isReadOnly, 1922
- put, 1922, 1923
- setReadOnly, 1923
- slice, 1923
- decaf::internal::nio::IntArrayBuffer, 2056
  - ~IntArrayBuffer, 2060
  - array, 2060
  - arrayOffset, 2061
  - asReadOnlyBuffer, 2061
  - compact, 2061
  - duplicate, 2062
  - get, 2062, 2063
  - hasArray, 2063
  - IntArrayBuffer, 2059, 2060
  - isReadOnly, 2063
  - put, 2063, 2064
  - setReadOnly, 2064
  - slice, 2064
- decaf::internal::nio::LongArrayBuffer, 2434
  - ~LongArrayBuffer, 2438
  - array, 2438
  - arrayOffset, 2439
  - asReadOnlyBuffer, 2439
  - compact, 2439
  - duplicate, 2440
  - get, 2440, 2441
  - hasArray, 2441
  - isReadOnly, 2441
  - LongArrayBuffer, 2437, 2438
  - put, 2441, 2442
  - setReadOnly, 2442
  - slice, 2442
- decaf::internal::nio::ShortArrayBuffer, 3449
  - ~ShortArrayBuffer, 3453
  - array, 3453
  - arrayOffset, 3454
  - asReadOnlyBuffer, 3454
  - compact, 3454
  - duplicate, 3455
  - get, 3455, 3456
  - hasArray, 3456
  - isReadOnly, 3456
  - put, 3456, 3457
  - setReadOnly, 3457
  - ShortArrayBuffer, 3452, 3453
  - slice, 3457
- decaf::internal::security, 151
- decaf::internal::security::SecureRandomImpl, 3336
  - ~SecureRandomImpl, 3337
  - providerGenerateSeed, 3337
  - providerNextBytes, 3337, 3338
  - providerSetSeed, 3338
  - SecureRandomImpl, 3337
- decaf::internal::util, 152
- decaf::internal::util::ByteArrayAdapter, 969
  - ~ByteArrayAdapter, 976
  - ByteArrayAdapter, 973–976
  - clear, 976
  - get, 976
  - getByteArray, 976
  - getCapacity, 977
  - getChar, 977
  - getCharArray, 977
  - getCharCapacity, 977
  - getDouble, 978
  - getDoubleArray, 978
  - getDoubleAt, 978
  - getDoubleCapacity, 978
  - getFloat, 979
  - getFloatArray, 979
  - getFloatAt, 979
  - getFloatCapacity, 980
  - getInt, 980
  - getIntArray, 980
  - getIntAt, 980
  - getIntCapacity, 981
  - getLong, 981
  - getLongArray, 981
  - getLongAt, 981
  - getLongCapacity, 982
  - getShort, 982
  - getShortArray, 982
  - getShortAt, 983
  - getShortCapacity, 983
  - operator[], 983
  - put, 984
  - putChar, 984
  - putDouble, 984
  - putDoubleAt, 985
  - putFloat, 985
  - putFloatAt, 986
  - putInt, 986
  - putIntAt, 986
  - putLong, 987
  - putLongAt, 987
  - putShort, 988
  - putShortAt, 988
  - read, 988
  - resize, 989
  - write, 989
- decaf::internal::util::concurrent, 153
- decaf::internal::util::concurrent::ConditionImpl, 1257
  - create, 1257

- destroy, 1257
- notify, 1257
- notifyAll, 1258
- wait, 1258
- decaf::internal::util::concurrent::MutexImpl, 2788
  - create, 2788
  - destroy, 2788
  - lock, 2788
  - trylock, 2789
  - unlock, 2789
- decaf::internal::util::concurrent::SynchronizableImpl, 3711
  - ~SynchronizableImpl, 3712
  - lock, 3712
  - notify, 3712
  - notifyAll, 3712
  - SynchronizableImpl, 3712
  - tryLock, 3713
  - unlock, 3713
  - wait, 3713, 3714
- decaf::internal::util::concurrent::Transferer, 3877
- decaf::internal::util::concurrent::TransferQueue, 3878
  - ~TransferQueue, 3878
  - transfer, 3879
  - TransferQueue, 3878
- decaf::internal::util::concurrent::TransferStack, 3881
  - ~TransferStack, 3881
  - transfer, 3881, 3882
  - TransferStack, 3881
- decaf::internal::util::GenericResource, 1974
  - ~GenericResource, 1974
  - GenericResource, 1974
  - getManaged, 1974
  - setManaged, 1974
- decaf::internal::util::HexStringParser, 1982
  - ~HexStringParser, 1982
  - HexStringParser, 1982
  - parse, 1982
  - parseDouble, 1982
  - parseFloat, 1983
- decaf::internal::util::Resource, 3280
  - ~Resource, 3280
- decaf::internal::util::ResourceLifecycleManager, 3281
  - ~ResourceLifecycleManager, 3281
  - addResource, 3281
  - destroyResources, 3281
  - ResourceLifecycleManager, 3281
- decaf::internal::util::TimerTaskHeap, 3804
  - ~TimerTaskHeap, 3805
  - adjustMinimum, 3805
  - decaf::util::TimerTask, 3803
  - deleteIfCancelled, 3805
  - find, 3805
  - insert, 3805
  - isEmpty, 3805
  - peek, 3805
  - remove, 3806
  - reset, 3806
  - size, 3806
  - TimerTaskHeap, 3805
- decaf::io, 154
  - decaf::io::BlockingByteArrayInputStream, 839
    - ~BlockingByteArrayInputStream, 840
    - available, 840
    - BlockingByteArrayInputStream, 840
    - close, 841
    - doReadArrayBounded, 841
    - doReadByte, 841
    - setByteArray, 841
    - skip, 841
  - decaf::io::BufferedInputStream, 934
    - ~BufferedInputStream, 936
    - available, 936
    - BufferedInputStream, 936
    - close, 937
    - doReadArrayBounded, 937
    - doReadByte, 937
    - mark, 937
    - markSupported, 937
    - reset, 938
    - skip, 938
  - decaf::io::BufferedOutputStream, 940
    - ~BufferedOutputStream, 941
    - BufferedOutputStream, 940
    - doWriteArray, 941
    - doWriteArrayBounded, 941
    - doWriteByte, 941
    - flush, 941
  - decaf::io::ByteArrayInputStream, 1020
    - ~ByteArrayInputStream, 1023
    - available, 1023
    - ByteArrayInputStream, 1022, 1023
    - doReadArrayBounded, 1024
    - doReadByte, 1024
    - mark, 1024
    - markSupported, 1024
    - reset, 1025
    - setByteArray, 1025, 1026
    - skip, 1026
  - decaf::io::ByteArrayOutputStream, 1028
    - ~ByteArrayOutputStream, 1029
    - ByteArrayOutputStream, 1028
    - doWriteArrayBounded, 1029

- doWriteByte, 1029
- reset, 1029
- size, 1029
- toByteArray, 1029
- toString, 1030
- writeTo, 1030
- decaf::io::Closeable, 1149
  - ~Closeable, 1149
  - close, 1149
- decaf::io::DataInput, 1558
  - ~DataInput, 1559
  - readBoolean, 1559
  - readByte, 1560
  - readChar, 1560
  - readDouble, 1560
  - readFloat, 1560
  - readFully, 1561
  - readInt, 1562
  - readLine, 1562
  - readLong, 1563
  - readShort, 1563
  - readString, 1563
  - readUnsignedByte, 1564
  - readUnsignedShort, 1564
  - readUTF, 1564
  - skipBytes, 1565
- decaf::io::DataInputStream, 1566
  - ~DataInputStream, 1567
  - DataInputStream, 1567
  - readBoolean, 1568
  - readByte, 1568
  - readChar, 1568
  - readDouble, 1568
  - readFloat, 1569
  - readFully, 1569, 1570
  - readInt, 1570
  - readLine, 1570
  - readLong, 1571
  - readShort, 1571
  - readString, 1572
  - readUnsignedByte, 1572
  - readUnsignedShort, 1572
  - readUTF, 1572
  - skipBytes, 1573
- decaf::io::DataOutput, 1574
  - ~DataOutput, 1575
  - writeBoolean, 1575
  - writeByte, 1575
  - writeBytes, 1575
  - writeChar, 1576
  - writeChars, 1576
  - writeDouble, 1576
  - writeFloat, 1577
  - writeInt, 1577
  - writeLong, 1577
  - writeShort, 1577
  - writeUnsignedShort, 1578
  - writeUTF, 1578
- decaf::io::DataOutputStream, 1579
  - ~DataOutputStream, 1580
  - buffer, 1582
  - DataOutputStream, 1580
  - doWriteArrayBounded, 1580
  - doWriteByte, 1580
  - size, 1581
  - writeBoolean, 1581
  - writeByte, 1582
  - writeBytes, 1582
  - writeChar, 1582
  - writeChars, 1582
  - writeDouble, 1582
  - writeFloat, 1582
  - writeInt, 1582
  - writeLong, 1582
  - writeShort, 1582
  - writeUnsignedShort, 1582
  - writeUTF, 1582
  - written, 1582
- decaf::io::EOFException, 1825
  - ~EOFException, 1826
  - clone, 1827
  - EOFException, 1825, 1826
- decaf::io::FileDescriptor, 1891
  - ~FileDescriptor, 1892
  - descriptor, 1892
  - err, 1892
  - FileDescriptor, 1892
  - in, 1892
  - out, 1892
  - readonly, 1892
  - sync, 1892
  - valid, 1892
- decaf::io::FilterInputStream, 1894
  - ~FilterInputStream, 1896
  - available, 1896
  - close, 1896
  - closed, 1899
  - doReadArray, 1897
  - doReadArrayBounded, 1897
  - doReadByte, 1897
  - FilterInputStream, 1896
  - inputStream, 1899
  - isClosed, 1897
  - mark, 1897
  - markSupported, 1898
  - own, 1899
  - reset, 1898
  - skip, 1899

- decaf::io::FilterOutputStream, 1900
  - ~FilterOutputStream, 1901
  - close, 1901
  - closed, 1903
  - doWriteArray, 1901
  - doWriteArrayBounded, 1902
  - doWriteByte, 1902
  - FilterOutputStream, 1901
  - flush, 1902
  - isClosed, 1902
  - outputStream, 1903
  - own, 1903
  - toString, 1902
- decaf::io::Flushable, 1936
  - ~Flushable, 1936
  - flush, 1936
- decaf::io::InputStream, 2043
  - ~InputStream, 2045
  - available, 2045
  - close, 2045
  - doReadArray, 2045
  - doReadArrayBounded, 2045
  - doReadByte, 2046
  - InputStream, 2045
  - lock, 2046
  - mark, 2046
  - markSupported, 2046
  - notify, 2047
  - notifyAll, 2047
  - read, 2047–2049
  - reset, 2049
  - skip, 2050
  - toString, 2050
  - tryLock, 2051
  - unlock, 2051
  - wait, 2051, 2052
- decaf::io::InputStreamReader, 2053
  - ~InputStreamReader, 2054
  - checkClosed, 2054
  - close, 2054
  - doReadArrayBounded, 2054
  - InputStreamReader, 2054
  - ready, 2054
- decaf::io::InterruptedIOException, 2127
  - ~InterruptedIOException, 2128
  - clone, 2129
  - InterruptedIOException, 2127, 2128
- decaf::io::IOException, 2142
  - ~IOException, 2143
  - clone, 2144
  - IOException, 2142, 2143
- decaf::io::OutputStream, 2907
  - ~OutputStream, 2909
  - close, 2909
  - doWriteArray, 2909
  - doWriteArrayBounded, 2909
  - doWriteByte, 2909
  - flush, 2910
  - lock, 2910
  - notify, 2910
  - notifyAll, 2910
  - OutputStream, 2909
  - toString, 2911
  - tryLock, 2911
  - unlock, 2911
  - wait, 2911, 2912
  - write, 2913, 2914
- decaf::io::OutputStreamWriter, 2915
  - ~OutputStreamWriter, 2916
  - checkClosed, 2916
  - close, 2916
  - doWriteArrayBounded, 2916
  - flush, 2916
  - OutputStreamWriter, 2915
- decaf::io::PushbackInputStream, 3141
  - ~PushbackInputStream, 3143
  - available, 3143
  - doReadArrayBounded, 3144
  - doReadByte, 3144
  - mark, 3144
  - markSupported, 3144
  - PushbackInputStream, 3143
  - reset, 3144
  - skip, 3145
  - unread, 3146
- decaf::io::Reader, 3163
  - ~Reader, 3164
  - doReadArray, 3164
  - doReadArrayBounded, 3164
  - doReadChar, 3164
  - doReadCharBuffer, 3165
  - doReadVector, 3165
  - mark, 3165
  - markSupported, 3165
  - read, 3165–3167
  - Reader, 3164
  - ready, 3167
  - reset, 3168
  - skip, 3168
- decaf::io::UnsupportedEncodingException, 3913
  - ~UnsupportedEncodingException, 3915
  - clone, 3915
  - UnsupportedEncodingException, 3913, 3914
- decaf::io::UTFDataFormatException, 3966
  - ~UTFDataFormatException, 3968
  - clone, 3968

- UTFDataFormatException, 3966, 3967
- decaf::io::Writer, 4021
  - ~Writer, 4022
  - append, 4022, 4023
  - doAppendChar, 4023
  - doAppendCharSequence, 4024
  - doAppendCharSequenceStartEnd, 4024
  - doWriteArray, 4024
  - doWriteArrayBounded, 4024
  - doWriteChar, 4024
  - doWriteString, 4024
  - doWriteStringBounded, 4024
  - doWriteVector, 4024
  - write, 4024–4026
  - Writer, 4022
- decaf::lang, 156
  - operator==, 158
- decaf::lang::Appendable, 725
  - ~Appendable, 726
  - append, 726
- decaf::lang::ArrayPointer, 730
  - ~ArrayPointer, 733
  - ArrayPointer, 732, 733
  - clone, 733
  - ConstReferenceType, 732
  - CounterType, 732
  - get, 733
  - length, 733
  - operator=, 734
  - operator==, 734, 736
  - operator[], 735
  - PointerType, 732
  - ReferenceType, 732
  - release, 735
  - reset, 735
  - swap, 736
- decaf::lang::ArrayPointerComparator, 737
  - compare, 737
  - operator(), 737
- decaf::lang::Boolean, 850
  - ~Boolean, 851
  - \_FALSE, 854
  - \_TRUE, 854
  - Boolean, 851
  - booleanValue, 851
  - compareTo, 851
  - equals, 852
  - operator<, 852
  - operator==, 853
  - parseBoolean, 853
  - toString, 853, 854
  - valueOf, 854
- decaf::lang::Byte, 960
  - ~Byte, 962
- Byte, 961
  - byteValue, 962
  - compareTo, 962
  - decode, 963
  - doubleValue, 963
  - equals, 963
  - floatValue, 963
  - intValue, 964
  - longValue, 964
  - MAX\_VALUE, 968
  - MIN\_VALUE, 968
  - operator<, 964
  - operator==, 965
  - parseByte, 965
  - shortValue, 966
  - SIZE, 968
  - toString, 966
  - valueOf, 966, 967
- decaf::lang::Character, 1100
  - byteValue, 1102
  - Character, 1102
  - compareTo, 1102
  - digit, 1102
  - doubleValue, 1103
  - equals, 1103
  - floatValue, 1103
  - intValue, 1103
  - isDigit, 1104
  - isISOControl, 1104
  - isLetter, 1104
  - isLetterOrDigit, 1104
  - isLowerCase, 1104
  - isUpperCase, 1104
  - isWhitespace, 1104
  - longValue, 1104
  - MAX\_RADIX, 1106
  - MAX\_VALUE, 1106
  - MIN\_RADIX, 1106
  - MIN\_VALUE, 1107
  - operator<, 1105
  - operator==, 1105
  - shortValue, 1106
  - SIZE, 1107
  - toString, 1106
  - valueOf, 1106
- decaf::lang::CharSequence, 1135
  - ~CharSequence, 1135
  - charAt, 1135
  - length, 1136
  - subSequence, 1136
  - toString, 1136
- decaf::lang::Comparable, 1214
  - ~Comparable, 1214
  - compareTo, 1214

- equals, 1215
- operator<, 1215
- operator==, 1215
- decaf::lang::Double, 1788
  - ~Double, 1790
  - byteValue, 1790
  - compare, 1790
  - compareTo, 1791
  - Double, 1790
  - doubleToLongBits, 1791
  - doubleToRawLongBits, 1792
  - doubleValue, 1792
  - equals, 1792, 1793
  - floatValue, 1793
  - intValue, 1793
  - isInfinite, 1793
  - isNaN, 1794
  - longBitsToDouble, 1794
  - longValue, 1794
  - MAX\_VALUE, 1798
  - MIN\_VALUE, 1798
  - NaN, 1798
  - NEGATIVE\_INFINITY, 1798
  - operator<, 1794, 1795
  - operator==, 1795
  - parseDouble, 1795
  - POSITIVE\_INFINITY, 1798
  - shortValue, 1796
  - SIZE, 1798
  - toHexString, 1796
  - toString, 1797
  - valueOf, 1797
- decaf::lang::DYNAMIC\_CAST\_TOKEN, 1820
- decaf::lang::Exception, 1831
  - ~Exception, 1833
  - buildMessage, 1834
  - cause, 1837
  - clone, 1834
  - Exception, 1832, 1833
  - getCause, 1835
  - getMessage, 1835
  - getStackTrace, 1835
  - getStackTraceString, 1835
  - initCause, 1836
  - message, 1837
  - operator=, 1836
  - printStackTrace, 1836
  - setMark, 1836
  - setMessage, 1836
  - setStackTrace, 1837
  - stackTrace, 1837
  - what, 1837
- decaf::lang::exceptions, 159
  - decaf::lang::exceptions::ClassCastException, 1145
    - ~ClassCastException, 1146
    - ClassCastException, 1145, 1146
    - clone, 1147
  - decaf::lang::exceptions::IllegalArgumentException, 1991
    - ~IllegalArgumentException, 1992
    - clone, 1993
    - IllegalArgumentException, 1991, 1992
  - decaf::lang::exceptions::IllegalMonitorStateException, 1994
    - ~IllegalMonitorStateException, 1995
    - clone, 1996
    - IllegalMonitorStateException, 1994, 1995
  - decaf::lang::exceptions::IllegalStateException, 1998
    - ~IllegalStateException, 1999
    - clone, 2000
    - IllegalStateException, 1998, 1999
  - decaf::lang::exceptions::IllegalThreadStateException, 2001
    - ~IllegalThreadStateException, 2002
    - clone, 2003
    - IllegalThreadStateException, 2001, 2002
  - decaf::lang::exceptions::IndexOutOfBoundsException, 2008
    - ~IndexOutOfBoundsException, 2009
    - clone, 2010
    - IndexOutOfBoundsException, 2008, 2009
  - decaf::lang::exceptions::InterruptedException, 2124
    - ~InterruptedException, 2125
    - clone, 2126
    - InterruptedException, 2124, 2125
  - decaf::lang::exceptions::InvalidStateException, 2139
    - ~InvalidStateException, 2140
    - clone, 2141
    - InvalidStateException, 2139, 2140
  - decaf::lang::exceptions::NoSuchElementException, 2826
    - ~NoSuchElementException, 2827
    - clone, 2828
    - NoSuchElementException, 2826, 2827
  - decaf::lang::exceptions::NullPointerException, 2832
    - ~NullPointerException, 2833
    - clone, 2834
    - NullPointerException, 2832, 2833
  - decaf::lang::exceptions::NumberFormatException, 2838
    - ~NumberFormatException, 2840
    - clone, 2840



- NumberFormatException, 2838, 2839
- decaf::lang::exceptions::RuntimeException, 3328
  - ~RuntimeException, 3329
  - clone, 3330
  - RuntimeException, 3328, 3329
- decaf::lang::exceptions::UnsupportedOperationException, 3916
  - ~UnsupportedOperationException, 3917
  - clone, 3918
  - UnsupportedOperationException, 3916, 3917
- decaf::lang::Float, 1904
  - ~Float, 1906
  - byteValue, 1906
  - compare, 1906
  - compareTo, 1907
  - doubleValue, 1907
  - equals, 1907, 1908
  - Float, 1906
  - floatToIntBits, 1908
  - floatToRawIntBits, 1908
  - floatValue, 1909
  - intBitsToFloat, 1909
  - intValue, 1909
  - isInfinite, 1909, 1910
  - isNaN, 1910
  - longValue, 1910
  - MAX\_VALUE, 1914
  - MIN\_VALUE, 1914
  - NaN, 1914
  - NEGATIVE\_INFINITY, 1914
  - operator<, 1910, 1911
  - operator==, 1911
  - parseFloat, 1911
  - POSITIVE\_INFINITY, 1914
  - shortValue, 1912
  - SIZE, 1914
  - toHexString, 1912
  - toString, 1912, 1913
  - valueOf, 1913
- decaf::lang::Integer, 2077
  - ~Integer, 2080
  - bitCount, 2080
  - byteValue, 2080
  - compareTo, 2080
  - decode, 2081
  - doubleValue, 2081
  - equals, 2081, 2082
  - floatValue, 2082
  - highestOneBit, 2082
  - Integer, 2079
  - intValue, 2082
  - longValue, 2082
  - lowestOneBit, 2083
  - MAX\_VALUE, 2090
  - MIN\_VALUE, 2090
  - numberOfLeadingZeros, 2083
  - numberOfTrailingZeros, 2083
  - operator<, 2084
  - operator==, 2084
  - parseInt, 2085
  - reverse, 2086
  - reverseBytes, 2086
  - rotateLeft, 2086
  - rotateRight, 2086
  - shortValue, 2087
  - signum, 2087
  - SIZE, 2090
  - toBinaryString, 2087
  - toHexString, 2088
  - toOctalString, 2088
  - toString, 2088, 2089
  - valueOf, 2089, 2090
- decaf::lang::Iterable, 2152
  - ~Iterable, 2152
  - iterator, 2152
- decaf::lang::Long, 2420
  - ~Long, 2423
  - bitCount, 2423
  - byteValue, 2423
  - compareTo, 2423
  - decode, 2424
  - doubleValue, 2424
  - equals, 2424, 2425
  - floatValue, 2425
  - highestOneBit, 2425
  - intValue, 2425
  - Long, 2422
  - longValue, 2425
  - lowestOneBit, 2426
  - MAX\_VALUE, 2433
  - MIN\_VALUE, 2433
  - numberOfLeadingZeros, 2426
  - numberOfTrailingZeros, 2426
  - operator<, 2427
  - operator==, 2427
  - parseLong, 2428
  - reverse, 2428
  - reverseBytes, 2429
  - rotateLeft, 2429
  - rotateRight, 2429
  - shortValue, 2430
  - signum, 2430
  - SIZE, 2433
  - toBinaryString, 2430
  - toHexString, 2430
  - toOctalString, 2431

- toString, 2431
- valueOf, 2432
- decaf::lang::Math, 2497
  - ~Math, 2499
  - abs, 2499, 2500
  - ceil, 2500
  - E, 2511
  - floor, 2501
  - Math, 2499
  - max, 2501, 2502
  - min, 2503, 2504
  - PI, 2511
  - pow, 2505
  - random, 2505
  - round, 2506
  - signum, 2506, 2507
  - sqrt, 2508
  - toDegrees, 2511
  - toRadians, 2511
- decaf::lang::Number, 2835
  - ~Number, 2835
  - byteValue, 2835
  - doubleValue, 2836
  - floatValue, 2836
  - intValue, 2836
  - longValue, 2836
  - shortValue, 2837
- decaf::lang::Pointer, 2946
  - ~Pointer, 2949
  - CounterType, 2948
  - dynamicCast, 2950
  - get, 2950
  - operator\*, 2950
  - operator->, 2951
  - operator=, 2951
  - operator==, 2951, 2953
  - Pointer, 2948, 2949
  - PointerType, 2948
  - ReferenceType, 2948
  - release, 2951
  - reset, 2952
  - staticCast, 2952
  - swap, 2952
- decaf::lang::PointerComparator, 2954
  - compare, 2954
  - operator(), 2954
- decaf::lang::Readable, 3160
  - ~Readable, 3160
  - read, 3160
- decaf::lang::Runnable, 3325
  - ~Runnable, 3325
  - run, 3325
- decaf::lang::Runtime, 3326
  - ~Runtime, 3326
- decaf::lang::System, 3733
- decaf::lang::Thread, 3773
- decaf::util::logging::LogManager, 2412
- getRuntime, 3326
- initializeRuntime, 3326
- shutdownRuntime, 3327
- decaf::lang::Short, 3440
  - ~Short, 3442
  - byteValue, 3442
  - compareTo, 3442
  - decode, 3443
  - doubleValue, 3443
  - equals, 3443
  - floatValue, 3443
  - intValue, 3444
  - longValue, 3444
  - MAX\_VALUE, 3448
  - MIN\_VALUE, 3448
  - operator<, 3444
  - operator==, 3445
  - parseShort, 3445
  - reverseBytes, 3446
  - Short, 3441
  - shortValue, 3446
  - SIZE, 3448
  - toString, 3446
  - valueOf, 3447
- decaf::lang::STATIC\_CAST\_TOKEN, 3587
- decaf::lang::String, 3665
  - ~String, 3666
  - charAt, 3666
  - isEmpty, 3666
  - length, 3667
  - String, 3666
  - subSequence, 3667
  - toString, 3667
- decaf::lang::System, 3726
  - ~System, 3728
  - arraycopy, 3728, 3729
  - availableProcessors, 3729
  - clearProperty, 3729
  - currentTimeMillis, 3730
  - decaf::lang::Runtime, 3733
  - getenv, 3730
  - getProperties, 3731
  - getProperty, 3731
  - nanoTime, 3732
  - setenv, 3732
  - setProperty, 3732
  - System, 3728
  - unsetenv, 3732
- decaf::lang::Thread, 3765
  - ~Thread, 3769
  - BLOCKED, 3768

- currentThread, 3769
- decaf::lang::Runtime, 3773
- decaf::util::concurrent::locks::LockSupport, 3773
  - getId, 3769
  - getName, 3769
  - getPriority, 3769
  - getState, 3769
  - getUncaughtExceptionHandler, 3770
  - isAlive, 3770
  - join, 3770, 3771
  - MAX\_PRIORITY, 3773
  - MIN\_PRIORITY, 3773
  - NEW, 3768
  - NORM\_PRIORITY, 3773
  - run, 3771
  - RUNNABLE, 3768
  - setName, 3771
  - setPriority, 3771
  - setUncaughtExceptionHandler, 3771
  - sleep, 3772
  - SLEEPING, 3768
  - start, 3772
  - State, 3768
  - TERMINATED, 3768
  - Thread, 3768
  - TIMED\_WAITING, 3768
  - toString, 3773
  - WAITING, 3768
  - yield, 3773
- decaf::lang::Thread::UncaughtExceptionHandler, 3906
  - ~UncaughtExceptionHandler, 3906
  - uncaughtException, 3906
- decaf::lang::ThreadGroup, 3776
  - ~ThreadGroup, 3776
  - ThreadGroup, 3776
- decaf::lang::Throwable, 3783
  - ~Throwable, 3784
  - clone, 3784
  - getCause, 3785
  - getMessage, 3785
  - getStackTrace, 3785
  - getStackTraceString, 3785
  - initCause, 3786
  - printStackTrace, 3786
  - setMark, 3786
  - Throwable, 3784
- decaf::net, 160
- decaf::net::BindException, 836
  - ~BindException, 837
  - BindException, 836, 837
  - clone, 838
- decaf::net::ConnectException, 1259
  - ~ConnectException, 1260
  - clone, 1261
  - ConnectException, 1259, 1260
- decaf::net::HttpRetryException, 1986
  - ~HttpRetryException, 1987
  - clone, 1988
  - HttpRetryException, 1986, 1987
- decaf::net::Inet4Address, 2011
  - ~Inet4Address, 2012
  - Inet4Address, 2012
  - InetAddress, 2014
  - isAnyLocalAddress, 2012
  - isLinkLocalAddress, 2012
  - isLoopbackAddress, 2012
  - isMCGlobal, 2012
  - isMCLinkLocal, 2013
  - isMCNodeLocal, 2013
  - isMCOrgLocal, 2013
  - isMCSiteLocal, 2013
  - isMulticastAddress, 2013
  - isSiteLocalAddress, 2014
- decaf::net::Inet6Address, 2015
  - ~Inet6Address, 2015
  - Inet6Address, 2015
  - InetAddress, 2015
- decaf::net::InetAddress, 2016
  - ~InetAddress, 2018
  - addressBytes, 2022
  - anyBytes, 2022
  - bytesToInt, 2018
  - getAddress, 2018
  - getAnyAddress, 2018
  - getByAddress, 2019
  - getHostAddress, 2019
  - getHostName, 2019
  - getLocalHost, 2020
  - getLoopbackAddress, 2020
  - hostname, 2022
  - InetAddress, 2018
  - isAnyLocalAddress, 2020
  - isLinkLocalAddress, 2020
  - isLoopbackAddress, 2020
  - isMCGlobal, 2020
  - isMCLinkLocal, 2021
  - isMCNodeLocal, 2021
  - isMCOrgLocal, 2021
  - isMCSiteLocal, 2021
  - isMulticastAddress, 2021
  - isSiteLocalAddress, 2022
  - loopbackBytes, 2022
  - reached, 2022
  - toString, 2022
- decaf::net::InetSocketAddress, 2023
  - ~InetSocketAddress, 2023

- InetAddress, 2023
- decaf::net::MalformedURLException, 2456
  - ~MalformedURLException, 2457
  - clone, 2458
  - MalformedURLException, 2456, 2457
- decaf::net::NoRouteToHostException, 2820
  - ~NoRouteToHostException, 2821
  - clone, 2822
  - NoRouteToHostException, 2820, 2821
- decaf::net::PortUnreachableException, 2973
  - ~PortUnreachableException, 2974
  - clone, 2975
  - PortUnreachableException, 2973, 2974
- decaf::net::ProtocolException, 3137
  - ~ProtocolException, 3138
  - clone, 3139
  - ProtocolException, 3137, 3138
- decaf::net::ServerSocket, 3352
  - ~ServerSocket, 3355
  - accept, 3356
  - bind, 3356
  - checkClosed, 3357
  - close, 3357
  - ensureCreated, 3357
  - getDefaultBacklog, 3357
  - getLocalPort, 3357
  - getReceiveBufferSize, 3357
  - getReuseAddress, 3357
  - getSoTimeout, 3358
  - implAccept, 3358
  - isBound, 3358
  - isClosed, 3358
  - ServerSocket, 3354, 3355
  - setReceiveBufferSize, 3358
  - setReuseAddress, 3359
  - setSocketImplFactory, 3359
  - setSoTimeout, 3359
  - setupSocketImpl, 3360
  - toString, 3360
- decaf::net::ServerSocketFactory, 3361
  - ~ServerSocketFactory, 3362
  - createServerSocket, 3362, 3363
  - getDefault, 3363
  - ServerSocketFactory, 3362
- decaf::net::Socket, 3503
  - ~Socket, 3508
  - accepted, 3508
  - bind, 3508
  - checkClosed, 3509
  - close, 3509
  - connect, 3509
  - ensureCreated, 3510
  - getInetAddress, 3510
  - getInputStream, 3510
  - getKeepAlive, 3510
  - getLocalAddress, 3510
  - getLocalPort, 3511
  - getOOBInline, 3511
  - getOutputStream, 3511
  - getPort, 3511
  - getReceiveBufferSize, 3511
  - getReuseAddress, 3512
  - getSendBufferSize, 3512
  - getSoLinger, 3512
  - getSoTimeout, 3512
  - getTcpNoDelay, 3513
  - getTrafficClass, 3513
  - impl, 3518
  - initSocketImpl, 3513
  - isBound, 3513
  - isClosed, 3513
  - isConnected, 3514
  - isInputShutdown, 3514
  - isOutputShutdown, 3514
  - sendUrgentData, 3514
  - ServerSocket, 3518
  - setKeepAlive, 3514
  - setOOBInline, 3514
  - setReceiveBufferSize, 3515
  - setReuseAddress, 3515
  - setSendBufferSize, 3515
  - setSocketImplFactory, 3516
  - setSoLinger, 3516
  - setSoTimeout, 3516
  - setTcpNoDelay, 3517
  - setTrafficClass, 3517
  - shutdownInput, 3517
  - shutdownOutput, 3517
  - Socket, 3506–3508
  - toString, 3518
- decaf::net::SocketAddress, 3519
  - ~SocketAddress, 3519
- decaf::net::SocketError, 3520
  - getErrorCode, 3520
  - getErrorString, 3520
- decaf::net::SocketException, 3521
  - ~SocketException, 3522
  - clone, 3522
  - SocketException, 3521, 3522
- decaf::net::SocketFactory, 3524
  - ~SocketFactory, 3525
  - createSocket, 3525–3527
  - getDefault, 3527
  - SocketFactory, 3525
- decaf::net::SocketImpl, 3529
  - ~SocketImpl, 3531
  - accept, 3531
  - address, 3536

- available, 3531
- bind, 3531
- close, 3532
- connect, 3532
- create, 3532
- fd, 3536
- getFileDescriptor, 3532
- getInetAddress, 3533
- getInputStream, 3533
- getLocalAddress, 3533
- getLocalPort, 3533
- getOption, 3533
- getOutputStream, 3534
- getPort, 3534
- listen, 3534
- localPort, 3536
- port, 3536
- sendUrgentData, 3535
- setOption, 3535
- shutdownInput, 3535
- shutdownOutput, 3535
- SocketImpl, 3531
- supportsUrgentData, 3536
- toString, 3536
- decaf::net::SocketImplFactory, 3537
  - ~SocketImplFactory, 3537
  - createSocketImpl, 3537
- decaf::net::SocketOptions, 3538
  - ~SocketOptions, 3539
  - SOCKET\_OPTION\_BINDADDR, 3539
  - SOCKET\_OPTION\_BROADCAST, 3539
  - SOCKET\_OPTION\_IP\_-  
MULTICAST\_IF, 3539
  - SOCKET\_OPTION\_IP\_-  
MULTICAST\_IF2, 3539
  - SOCKET\_OPTION\_IP\_-  
MULTICAST\_LOOP, 3540
  - SOCKET\_OPTION\_IP\_TOS, 3540
  - SOCKET\_OPTION\_KEEPAIVE, 3540
  - SOCKET\_OPTION\_LINGER, 3540
  - SOCKET\_OPTION\_OOINLINE, 3540
  - SOCKET\_OPTION\_RCVBUF, 3540
  - SOCKET\_OPTION\_REUSEADDR, 3541
  - SOCKET\_OPTION\_SNDBUF, 3541
  - SOCKET\_OPTION\_TCP\_NODELAY, 3541
  - SOCKET\_OPTION\_TIMEOUT, 3541
- decaf::net::SocketTimeoutException, 3542
  - ~SocketTimeoutException, 3543
  - clone, 3544
  - SocketTimeoutException, 3542, 3543
- decaf::net::ssl, 162
  - decaf::net::ssl::SSLContext, 3545
    - ~SSLContext, 3545
    - getDefault, 3545
    - getDefaultSSLParameters, 3546
    - getServerSocketFactory, 3546
    - getSocketFactory, 3546
    - getSupportedSSLParameters, 3546
    - setDefault, 3547
    - SSLContext, 3545
  - decaf::net::ssl::SSLContextSpi, 3548
    - ~SSLContextSpi, 3548
    - providerGetDefaultSSLParameters, 3548
    - providerGetServerSocketFactory, 3549
    - providerGetSocketFactory, 3549
    - providerGetSupportedSSLParameters, 3549
    - providerInit, 3550
  - decaf::net::ssl::SSLParameters, 3551
    - ~SSLParameters, 3552
    - getCipherSuites, 3552
    - getNeedClientAuth, 3552
    - getProtocols, 3552
    - getWantClientAuth, 3552
    - setCipherSuites, 3552
    - setNeedClientAuth, 3553
    - setProtocols, 3553
    - setWantClientAuth, 3553
    - SSLParameters, 3551, 3552
  - decaf::net::ssl::SSLServerSocket, 3554
    - ~SSLServerSocket, 3556
    - getEnabledCipherSuites, 3556
    - getEnabledProtocols, 3557
    - getNeedClientAuth, 3557
    - getSupportedCipherSuites, 3557
    - getSupportedProtocols, 3557
    - getWantClientAuth, 3557
    - setEnabledCipherSuites, 3558
    - setEnabledProtocols, 3558
    - setNeedClientAuth, 3558
    - setWantClientAuth, 3558
    - SSLServerSocket, 3555, 3556
  - decaf::net::ssl::SSLServerSocketFactory, 3560
    - ~SSLServerSocketFactory, 3561
    - getDefault, 3561
    - getDefaultCipherSuites, 3561
    - getSupportedCipherSuites, 3561
    - SSLServerSocketFactory, 3561
  - decaf::net::ssl::SSLSocket, 3563
    - ~SSLSocket, 3566
    - getEnabledCipherSuites, 3566
    - getEnabledProtocols, 3566
    - getNeedClientAuth, 3566
    - getSSLParameters, 3567
    - getSupportedCipherSuites, 3567

- getSupportedProtocols, 3567
- getUseClientMode, 3567
- getWantClientAuth, 3568
- setEnabledCipherSuites, 3568
- setEnabledProtocols, 3568
- setNeedClientAuth, 3569
- setSSLParameters, 3569
- setUseClientMode, 3569
- setWantClientAuth, 3570
- SSLSocket, 3564, 3565
- startHandshake, 3570
- decaf::net::ssl::SSLSocketFactory, 3571
  - ~SSLSocketFactory, 3572
  - createSocket, 3572
  - getDefault, 3572
  - getDefaultCipherSuites, 3572
  - getSupportedCipherSuites, 3573
  - SSLSocketFactory, 3572
- decaf::net::UnknownHostException, 3907
  - ~UnknownHostException, 3908
  - clone, 3909
  - UnknownHostException, 3907, 3908
- decaf::net::UnknownServiceException, 3910
  - ~UnknownServiceException, 3911
  - clone, 3912
  - UnknownServiceException, 3910, 3911
- decaf::net::URI, 3921
  - ~URI, 3924
  - compareTo, 3925
  - create, 3925
  - equals, 3925
  - getAuthority, 3925
  - getFragment, 3925
  - getHost, 3925
  - getPath, 3926
  - getPort, 3926
  - getQuery, 3926
  - getRawAuthority, 3926
  - getRawFragment, 3926
  - getRawPath, 3926
  - getRawQuery, 3927
  - getRawSchemeSpecificPart, 3927
  - getRawUserInfo, 3927
  - getScheme, 3927
  - getSchemeSpecificPart, 3927
  - getUserInfo, 3927
  - isAbsolute, 3928
  - isOpaque, 3928
  - normalize, 3928
  - operator<, 3928
  - operator==, 3929
  - parseServerAuthority, 3929
  - relativize, 3929
  - resolve, 3930
  - toString, 3931
  - toURL, 3931
  - URI, 3923, 3924
- decaf::net::URISyntaxException, 3948
  - ~URISyntaxException, 3950
  - clone, 3950
  - getIndex, 3950
  - getInput, 3951
  - getReason, 3951
  - URISyntaxException, 3948–3950
- decaf::net::URL, 3960
  - ~URL, 3961
  - URL, 3961
- decaf::net::URLDecoder, 3962
  - ~URLDecoder, 3962
  - decode, 3962
- decaf::net::URLEncoder, 3963
  - ~URLEncoder, 3963
  - encode, 3963
- decaf::nio, 163
- decaf::nio::Buffer, 928
  - ~Buffer, 930
  - \_capacity, 933
  - \_limit, 933
  - \_mark, 933
  - \_markSet, 933
  - \_position, 933
  - Buffer, 930
  - capacity, 930
  - clear, 930
  - flip, 931
  - hasRemaining, 931
  - isReadOnly, 931
  - limit, 931, 932
  - mark, 932
  - position, 932
  - remaining, 932
  - reset, 933
  - rewind, 933
- decaf::nio::BufferOverflowException, 954
  - ~BufferOverflowException, 955
  - BufferOverflowException, 954, 955
  - clone, 956
- decaf::nio::BufferUnderflowException, 957
  - ~BufferUnderflowException, 958
  - BufferUnderflowException, 957, 958
  - clone, 959
- decaf::nio::ByteBuffer, 1031
  - ~ByteBuffer, 1036
  - allocate, 1036
  - array, 1037
  - arrayOffset, 1037
  - asCharBuffer, 1037
  - asDoubleBuffer, 1038

- asFloatBuffer, 1038
- asIntBuffer, 1038
- asLongBuffer, 1039
- asReadOnlyBuffer, 1039
- asShortBuffer, 1039
- ByteBuffer, 1036
- compact, 1040
- compareTo, 1040
- duplicate, 1040
- equals, 1040
- get, 1040–1042
- getChar, 1042
- getDouble, 1042, 1043
- getFloat, 1043
- getInt, 1044
- getLong, 1044, 1045
- getShort, 1045
- hasArray, 1046
- isReadOnly, 1046
- operator<, 1046
- operator==, 1046
- put, 1046–1048
- putChar, 1049
- putDouble, 1049, 1050
- putFloat, 1050, 1051
- putInt, 1051
- putLong, 1052
- putShort, 1053
- slice, 1053
- toString, 1054
- wrap, 1054
- decaf::nio::CharBuffer, 1119
  - ~CharBuffer, 1122
  - allocate, 1122
  - append, 1123
  - array, 1124
  - arrayOffset, 1124
  - asReadOnlyBuffer, 1125
  - charAt, 1125
  - CharBuffer, 1122
  - compact, 1125
  - compareTo, 1126
  - duplicate, 1126
  - equals, 1126
  - get, 1126, 1127
  - hasArray, 1128
  - length, 1128
  - operator<, 1128
  - operator==, 1128
  - put, 1128–1131
  - read, 1131
  - slice, 1132
  - subSequence, 1132
  - toString, 1133
  - wrap, 1133
- decaf::nio::DoubleBuffer, 1809
  - ~DoubleBuffer, 1811
  - allocate, 1811
  - array, 1812
  - arrayOffset, 1812
  - asReadOnlyBuffer, 1812
  - compact, 1813
  - compareTo, 1813
  - DoubleBuffer, 1811
  - duplicate, 1813
  - equals, 1814
  - get, 1814, 1815
  - hasArray, 1815
  - operator<, 1815
  - operator==, 1816
  - put, 1816, 1817
  - slice, 1818
  - toString, 1818
  - wrap, 1818, 1819
- decaf::nio::FloatBuffer, 1925
  - ~FloatBuffer, 1927
  - allocate, 1927
  - array, 1927
  - arrayOffset, 1928
  - asReadOnlyBuffer, 1928
  - compact, 1929
  - compareTo, 1929
  - duplicate, 1929
  - equals, 1929
  - FloatBuffer, 1927
  - get, 1929–1931
  - hasArray, 1931
  - operator<, 1931
  - operator==, 1931
  - put, 1931–1933
  - slice, 1933
  - toString, 1934
  - wrap, 1934
- decaf::nio::IntBuffer, 2066
  - ~IntBuffer, 2068
  - allocate, 2068
  - array, 2068
  - arrayOffset, 2069
  - asReadOnlyBuffer, 2069
  - compact, 2070
  - compareTo, 2070
  - duplicate, 2070
  - equals, 2070
  - get, 2070–2072
  - hasArray, 2072
  - IntBuffer, 2068
  - operator<, 2072
  - operator==, 2072

- put, 2072–2074
- slice, 2075
- toString, 2075
- wrap, 2075
- decaf::nio::InvalidMarkException, 2135
  - ~InvalidMarkException, 2136
  - clone, 2137
  - InvalidMarkException, 2135, 2136
- decaf::nio::LongBuffer, 2444
  - ~LongBuffer, 2446
  - allocate, 2446
  - array, 2447
  - arrayOffset, 2447
  - asReadOnlyBuffer, 2447
  - compact, 2448
  - compareTo, 2448
  - duplicate, 2448
  - equals, 2448
  - get, 2448–2450
  - hasArray, 2450
  - LongBuffer, 2446
  - operator<, 2450
  - operator==, 2450
  - put, 2450–2452
  - slice, 2453
  - toString, 2453
  - wrap, 2453
- decaf::nio::ReadOnlyBufferException, 3169
  - ~ReadOnlyBufferException, 3170
  - clone, 3171
  - ReadOnlyBufferException, 3169, 3170
- decaf::nio::ShortBuffer, 3459
  - ~ShortBuffer, 3461
  - allocate, 3461
  - array, 3461
  - arrayOffset, 3462
  - asReadOnlyBuffer, 3462
  - compact, 3463
  - compareTo, 3463
  - duplicate, 3463
  - equals, 3463
  - get, 3463–3465
  - hasArray, 3465
  - operator<, 3465
  - operator==, 3465
  - put, 3465–3467
  - ShortBuffer, 3461
  - slice, 3467
  - toString, 3468
  - wrap, 3468
- decaf::security, 164
- decaf::security::auth, 165
- decaf::security::auth::x500, 166
- decaf::security::auth::x500::X500Principal, 4027
  - ~X500Principal, 4027
  - getEncoded, 4027
  - getName, 4027
  - hashCode, 4027
- decaf::security::cert, 167
- decaf::security::cert::Certificate, 1086
  - ~Certificate, 1087
  - equals, 1087
  - getEncoded, 1087
  - getPublicKey, 1087
  - getType, 1087
  - toString, 1088
  - verify, 1088
- decaf::security::cert::CertificateEncodingException, 1090
  - ~CertificateEncodingException, 1091
  - CertificateEncodingException, 1090, 1091
  - clone, 1091
- decaf::security::cert::CertificateException, 1092
  - ~CertificateException, 1093
  - CertificateException, 1092, 1093
  - clone, 1093
- decaf::security::cert::CertificateExpiredException, 1094
  - ~CertificateExpiredException, 1095
  - CertificateExpiredException, 1094, 1095
  - clone, 1095
- decaf::security::cert::CertificateNotYetValidException, 1096
  - ~CertificateNotYetValidException, 1097
  - CertificateNotYetValidException, 1096, 1097
  - clone, 1097
- decaf::security::cert::CertificateParsingException, 1098
  - ~CertificateParsingException, 1099
  - CertificateParsingException, 1098, 1099
  - clone, 1099
- decaf::security::cert::X509Certificate, 4028
  - ~X509Certificate, 4029
  - checkValidity, 4029
  - getBasicConstraints, 4029
  - getIssuerUniqueID, 4029
  - getIssuerX500Principal, 4029
  - getKeyUsage, 4029
  - getNotAfter, 4029
  - getNotBefore, 4029
  - getSigAlgName, 4029
  - getSigAlgOID, 4029
  - getSigAlgParams, 4029
  - getSignature, 4029
  - getSubjectUniqueID, 4029
  - getSubjectX500Principal, 4029
  - getTBSCertificate, 4029



- getVersion, 4029
- decaf::security::GeneralSecurityException, 1971
  - ~GeneralSecurityException, 1972
  - clone, 1973
  - GeneralSecurityException, 1971, 1972
- decaf::security::InvalidKeyException, 2132
  - ~InvalidKeyException, 2133
  - clone, 2134
  - InvalidKeyException, 2132, 2133
- decaf::security::Key, 2293
  - ~Key, 2294
  - getAlgorithm, 2294
  - getEncoded, 2294
  - getFormat, 2294
- decaf::security::KeyException, 2295
  - ~KeyException, 2296
  - clone, 2297
  - KeyException, 2295, 2296
- decaf::security::KeyManagementException, 2298
  - ~KeyManagementException, 2299
  - clone, 2300
  - KeyManagementException, 2298, 2299
- decaf::security::NoSuchAlgorithmException, 2823
  - ~NoSuchAlgorithmException, 2824
  - clone, 2825
  - NoSuchAlgorithmException, 2823, 2824
- decaf::security::NoSuchProviderException, 2829
  - ~NoSuchProviderException, 2830
  - clone, 2831
  - NoSuchProviderException, 2829, 2830
- decaf::security::Principal, 3026
  - ~Principal, 3026
  - equals, 3026
  - getName, 3026
- decaf::security::PublicKey, 3140
  - ~PublicKey, 3140
- decaf::security::SecureRandom, 3331
  - ~SecureRandom, 3333
  - next, 3333
  - nextBytes, 3333, 3334
  - SecureRandom, 3332, 3333
  - setSeed, 3334, 3335
- decaf::security::SecureRandomSpi, 3339
  - ~SecureRandomSpi, 3339
  - providerGenerateSeed, 3339
  - providerNextBytes, 3340
  - providerSetSeed, 3340
  - SecureRandomSpi, 3339
- decaf::security::SignatureException, 3497
  - ~SignatureException, 3498
  - clone, 3499
  - SignatureException, 3497, 3498
- decaf::util, 168
- decaf::util::AbstractCollection, 179
  - ~AbstractCollection, 182
  - AbstractCollection, 182
  - add, 182
  - addAll, 182
  - clear, 183
  - contains, 184
  - containsAll, 184
  - copy, 185
  - equals, 185
  - isEmpty, 185
  - lock, 186
  - mutex, 191
  - notify, 186
  - notifyAll, 186
  - operator=, 187
  - remove, 187
  - removeAll, 188
  - retainAll, 188
  - toArray, 189
  - tryLock, 189
  - unlock, 190
  - wait, 190, 191
- decaf::util::AbstractList, 192
  - ~AbstractList, 192
- decaf::util::AbstractMap, 193
  - ~AbstractMap, 193
- decaf::util::AbstractQueue, 194
  - ~AbstractQueue, 195
  - AbstractQueue, 195
  - add, 195
  - addAll, 195
  - clear, 196
  - element, 196
  - remove, 196
- decaf::util::AbstractSequentialList, 198
  - ~AbstractSequentialList, 198
- decaf::util::AbstractSet, 199
  - ~AbstractSet, 199
  - removeAll, 199
- decaf::util::Collection, 1184
  - ~Collection, 1186
  - add, 1186
  - addAll, 1186
  - clear, 1187
  - contains, 1188
  - containsAll, 1189
  - equals, 1189
  - isEmpty, 1189
  - remove, 1190
  - removeAll, 1191
  - retainAll, 1191
  - size, 1191

- toArray, 1192
- decaf::util::Comparator, 1217
  - ~Comparator, 1217
  - compare, 1217
  - operator(), 1218
- decaf::util::comparators, 170
- decaf::util::comparators::Less, 2328
  - ~Less, 2328
  - compare, 2328
  - Less, 2328
  - operator(), 2329
- decaf::util::concurrent, 171
- decaf::util::concurrent::atomic, 173
- decaf::util::concurrent::atomic::AtomicBoolean, 738
  - ~AtomicBoolean, 738
  - AtomicBoolean, 738
  - compareAndSet, 739
  - get, 739
  - getAndSet, 739
  - set, 739
  - toString, 739
- decaf::util::concurrent::atomic::AtomicInteger, 741
  - ~AtomicInteger, 742
  - addAndGet, 742
  - AtomicInteger, 742
  - compareAndSet, 743
  - decrementAndGet, 743
  - doubleValue, 743
  - floatValue, 743
  - get, 743
  - getAndAdd, 744
  - getAndDecrement, 744
  - getAndIncrement, 744
  - getAndSet, 744
  - incrementAndGet, 744
  - intValue, 745
  - longValue, 745
  - set, 745
  - toString, 745
- decaf::util::concurrent::atomic::AtomicRefCounter, 746
  - ~AtomicRefCounter, 747
  - AtomicRefCounter, 747
  - release, 747
  - swap, 748
- decaf::util::concurrent::atomic::AtomicReference, 749
  - ~AtomicReference, 750
  - AtomicReference, 750
  - compareAndSet, 750
  - get, 750
  - getAndSet, 750
  - set, 750
  - toString, 751
- decaf::util::concurrent::BlockingQueue, 843
  - ~BlockingQueue, 846
  - drainTo, 846
  - offer, 847
  - poll, 847
  - put, 848
  - remainingCapacity, 848
  - take, 848
- decaf::util::concurrent::BrokenBarrierException, 860
  - ~BrokenBarrierException, 861
  - BrokenBarrierException, 860, 861
  - clone, 862
- decaf::util::concurrent::Callable, 1082
  - ~Callable, 1082
  - call, 1082
- decaf::util::concurrent::CancellationException, 1083
  - ~CancellationException, 1084
  - CancellationException, 1083, 1084
  - clone, 1085
- decaf::util::concurrent::ConcurrentMap, 1228
  - ~ConcurrentMap, 1229
  - putIfAbsent, 1229
  - remove, 1229
  - replace, 1230, 1231
- decaf::util::concurrent::ConcurrentStlMap, 1233
  - ~ConcurrentStlMap, 1237
  - clear, 1237
  - ConcurrentStlMap, 1237
  - containsKey, 1238
  - containsValue, 1238
  - copy, 1238, 1239
  - equals, 1239
  - get, 1239, 1240
  - isEmpty, 1240
  - keySet, 1240
  - lock, 1241
  - notify, 1241
  - notifyAll, 1241
  - put, 1242
  - putAll, 1242
  - putIfAbsent, 1243
  - remove, 1243, 1244
  - replace, 1244, 1245
  - size, 1245
  - tryLock, 1246
  - unlock, 1246
  - values, 1246
  - wait, 1246, 1247
- decaf::util::concurrent::ConditionHandle, 1255
  - ~ConditionHandle, 1255

- condition, 1255
- ConditionHandle, 1255
- criticalSection, 1255
- generation, 1255
- mutex, 1255
- numWaiting, 1255
- numWake, 1255
- semaphore, 1255
- decaf::util::concurrent::CountDownLatch, 1521
  - ~CountDownLatch, 1521
  - await, 1522, 1523
  - countDown, 1523
  - CountDownLatch, 1521
  - getCount, 1523
- decaf::util::concurrent::Delayed, 1720
  - ~Delayed, 1720
  - getDelay, 1720
- decaf::util::concurrent::ExecutionException, 1866
  - ~ExecutionException, 1867
  - clone, 1868
  - ExecutionException, 1866, 1867
- decaf::util::concurrent::Executor, 1869
  - ~Executor, 1870
  - execute, 1870
- decaf::util::concurrent::ExecutorService, 1871
  - ~ExecutorService, 1872
  - awaitTermination, 1872
- decaf::util::concurrent::Future, 1966
  - ~Future, 1967
  - cancel, 1967
  - get, 1967
  - isCancelled, 1968
  - isDone, 1968
- decaf::util::concurrent::Lock, 2375
  - ~Lock, 2375
  - isLocked, 2376
  - Lock, 2375
  - lock, 2376
  - unlock, 2376
- decaf::util::concurrent::locks, 174
- decaf::util::concurrent::locks::Condition, 1249
  - ~Condition, 1251
  - await, 1251
  - awaitNanos, 1252
  - awaitUninterruptibly, 1253
  - awaitUntil, 1254
  - signal, 1254
  - signalAll, 1254
- decaf::util::concurrent::locks::Lock, 2377
  - ~Lock, 2378
  - lock, 2378
  - lockInterruptibly, 2379
  - newCondition, 2379
  - tryLock, 2380, 2381
  - unlock, 2381
- decaf::util::concurrent::locks::LockSupport, 2383
  - ~LockSupport, 2384
- decaf::lang::Thread, 3773
  - park, 2384
  - parkNanos, 2384
  - parkUntil, 2385
  - unpark, 2385
- decaf::util::concurrent::locks::ReadWriteLock, 3172
  - ~ReadWriteLock, 3173
  - readLock, 3173
  - writeLock, 3173
- decaf::util::concurrent::locks::ReentrantLock, 3182
  - ~ReentrantLock, 3183
  - getHoldCount, 3183
  - isFair, 3184
  - isHeldByCurrentThread, 3184
  - isLocked, 3184
  - lock, 3184
  - lockInterruptibly, 3185
  - newCondition, 3185
  - ReentrantLock, 3183
  - toString, 3186
  - tryLock, 3186, 3187
  - unlock, 3188
- decaf::util::concurrent::Mutex, 2782
  - ~Mutex, 2783
  - lock, 2783
  - Mutex, 2783
  - notify, 2783
  - notifyAll, 2783
  - tryLock, 2784
  - unlock, 2784
  - wait, 2785, 2786
- decaf::util::concurrent::MutexHandle, 2787
  - ~MutexHandle, 2787
  - lock\_count, 2787
  - lock\_owner, 2787
  - mutex, 2787
  - MutexHandle, 2787
- decaf::util::concurrent::PooledThread, 2968
  - ~PooledThread, 2968
  - getPooledThreadListener, 2969
  - isBusy, 2969
  - PooledThread, 2968
  - run, 2969
  - setPooledThreadListener, 2969
  - stop, 2969
- decaf::util::concurrent::PooledThreadListener, 2971

- ~PooledThreadListener, 2971
- onTaskCompleted, 2971
- onTaskException, 2972
- onTaskStarted, 2972
- decaf::util::concurrent::RejectedExecutionException, 3189
  - ~RejectedExecutionException, 3190
  - clone, 3191
  - RejectedExecutionException, 3189, 3190
- decaf::util::concurrent::RejectedExecutionHandler, 3192
  - ~RejectedExecutionHandler, 3192
  - rejectedExecution, 3192
- decaf::util::concurrent::Semaphore, 3341
  - ~Semaphore, 3344
  - acquire, 3344
  - acquireUninterruptibly, 3345
  - availablePermits, 3345
  - drainPermits, 3346
  - isFair, 3346
  - release, 3346
  - Semaphore, 3343
  - toString, 3347
  - tryAcquire, 3347–3349
- decaf::util::concurrent::Synchronizable, 3700
  - ~Synchronizable, 3701
  - lock, 3701
  - notify, 3702
  - notifyAll, 3703
  - tryLock, 3704
  - unlock, 3705
  - wait, 3706, 3708, 3709
- decaf::util::concurrent::SynchronousQueue, 3716
  - ~SynchronousQueue, 3718
  - clear, 3718
  - contains, 3718
  - containsAll, 3719
  - drainTo, 3719, 3720
  - equals, 3720
  - isEmpty, 3720
  - iterator, 3721
  - offer, 3721
  - peek, 3722
  - poll, 3722
  - put, 3723
  - remainingCapacity, 3723
  - remove, 3723
  - removeAll, 3724
  - retainAll, 3724
  - size, 3724
  - SynchronousQueue, 3718
  - take, 3724
  - toArray, 3724
- decaf::util::concurrent::TaskListener, 3735
  - ~TaskListener, 3735
  - onTaskComplete, 3735
  - onTaskException, 3735
- decaf::util::concurrent::ThreadFactory, 3774
  - ~ThreadFactory, 3774
  - newThread, 3774
- decaf::util::concurrent::ThreadPool, 3777
  - ~ThreadPool, 3779
  - DEFAULT\_MAX\_BLOCK\_SIZE, 3782
  - DEFAULT\_MAX\_POOL\_SIZE, 3782
  - deQueueTask, 3779
  - getBacklog, 3779
  - getBlockSize, 3779
  - getFreeThreadCount, 3779
  - getInstance, 3780
  - getMaxThreads, 3780
  - getPoolSize, 3780
  - onTaskCompleted, 3780
  - onTaskException, 3780
  - onTaskStarted, 3781
  - queueTask, 3781
  - reserve, 3781
  - setBlockSize, 3781
  - setMaxThreads, 3781
  - Task, 3779
  - ThreadPool, 3779
- decaf::util::concurrent::TimeoutException, 3787
  - ~TimeoutException, 3788
  - clone, 3789
  - TimeoutException, 3787, 3788
- decaf::util::concurrent::TimeUnit, 3807
  - ~TimeUnit, 3809
  - compareTo, 3809
  - convert, 3810
  - DAYS, 3815
  - equals, 3810
  - HOURS, 3815
  - MICROSECONDS, 3815
  - MILLISECONDS, 3815
  - MINUTES, 3815
  - NANOSECONDS, 3815
  - operator<, 3810
  - operator==, 3810
  - SECONDS, 3815
  - sleep, 3811
  - timedJoin, 3811
  - timedWait, 3811
  - TimeUnit, 3809
  - toDays, 3812
  - toHours, 3812
  - toMicros, 3813
  - toMillis, 3813
  - toMinutes, 3813

- toNanos, 3814
- toSeconds, 3814
- toString, 3814
- valueOf, 3815
- values, 3816
- decaf::util::Date, 1665
  - ~Date, 1666
  - after, 1666
  - before, 1666
  - compareTo, 1667
  - Date, 1666
  - equals, 1667
  - getTime, 1667
  - operator<, 1667
  - operator=, 1667
  - operator==, 1668
  - setTime, 1668
  - toString, 1668
- decaf::util::Iterator, 2154
  - ~Iterator, 2154
  - hasNext, 2154
  - next, 2154
  - remove, 2155
- decaf::util::List, 2337
  - ~List, 2338
  - add, 2338
  - addAll, 2338
  - get, 2339
  - indexOf, 2339
  - lastIndexOf, 2340
  - List, 2338
  - listIterator, 2340, 2341
  - remove, 2342
  - set, 2342
- decaf::util::ListIterator, 2344
  - ~ListIterator, 2345
  - add, 2345
  - hasPrevious, 2345
  - nextIndex, 2345
  - previous, 2345
  - previousIndex, 2346
  - set, 2346
- decaf::util::logging, 175
  - Debug, 176
  - Error, 176
  - Fatal, 176
  - Info, 176
  - Levels, 176
  - Markblock, 176
  - Null, 176
  - Off, 176
  - Throwing, 176
  - Warn, 176
- decaf::util::logging::ConsoleHandler, 1399
  - ~ConsoleHandler, 1399
  - close, 1399
  - ConsoleHandler, 1399
  - publish, 1400
- decaf::util::logging::ErrorManager, 1828
  - ~ErrorManager, 1829
  - CLOSE\_FAILURE, 1829
  - error, 1829
  - ErrorManager, 1829
  - FLUSH\_FAILURE, 1829
  - FORMAT\_FAILURE, 1829
  - GENERIC\_FAILURE, 1829
  - OPEN\_FAILURE, 1829
  - WRITE\_FAILURE, 1829
- decaf::util::logging::Filter, 1893
  - ~Filter, 1893
  - isLoggable, 1893
- decaf::util::logging::Formatter, 1964
  - ~Formatter, 1964
  - format, 1964
  - formatMessage, 1965
  - getHead, 1965
  - getTail, 1965
- decaf::util::logging::Handler, 1978
  - ~Handler, 1979
  - flush, 1979
  - getErrorManager, 1979
  - getFilter, 1979
  - getFormatter, 1979
  - getLevel, 1979
  - Handler, 1979
  - isLoggable, 1980
  - publish, 1980
  - reportError, 1980
  - setErrorManager, 1980
  - setFilter, 1980
  - setFormatter, 1981
  - setLevel, 1981
- decaf::util::logging::Level, 2332
  - ~Level, 2334
  - ALL, 2335
  - compareTo, 2334
  - CONFIG, 2335
  - DEBUG, 2335
  - equals, 2334
  - FINE, 2335
  - FINER, 2335
  - FINEST, 2335
  - getName, 2334
  - INFO, 2336
  - INHERIT, 2336
  - intValue, 2334
  - Level, 2333
  - OFF, 2336

- operator<, 2334
- operator==, 2334
- parse, 2334
- SEVERE, 2336
- toString, 2335
- WARNING, 2336
- decaf::util::logging::Logger, 2386
  - ~Logger, 2389
  - addHandler, 2389
  - config, 2389
  - debug, 2390
  - entering, 2390
  - exiting, 2390
  - fine, 2390
  - finer, 2391
  - finest, 2391
  - getAnonymousLogger, 2391
  - getFilter, 2392
  - getHandlers, 2392
  - getLevel, 2392
  - getLogger, 2392
  - getName, 2392
  - getParent, 2393
  - getUseParentHandlers, 2393
  - info, 2393
  - isLoggable, 2393
  - log, 2394
  - Logger, 2389
  - removeHandler, 2395
  - setFilter, 2395
  - setLevel, 2395
  - setParent, 2395
  - setUseParentHandlers, 2395
  - severe, 2396
  - throwing, 2396
  - warning, 2396
- decaf::util::logging::LoggerHierarchy, 2398
  - ~LoggerHierarchy, 2398
  - LoggerHierarchy, 2398
- decaf::util::logging::LogManager, 2406
  - ~LogManager, 2408
  - addLogger, 2409
  - addPropertyChangeListener, 2409
  - decaf::lang::Runtime, 2412
  - getLogger, 2409
  - getLoggerNames, 2409
  - getLogManager, 2409
  - getProperties, 2410
  - getProperty, 2410
  - LogManager, 2408
  - operator=, 2410
  - readConfiguration, 2410, 2411
  - removePropertyChangeListener, 2411
  - reset, 2411
  - setProperties, 2411
- decaf::util::logging::LogRecord, 2413
  - ~LogRecord, 2414
  - getLevel, 2414
  - getLoggerName, 2414
  - getMessage, 2414
  - getSourceFile, 2415
  - getSourceFunction, 2415
  - getSourceLine, 2415
  - getThrown, 2415
  - getTimestamp, 2415
  - getTreadId, 2415
  - LogRecord, 2414
  - setLevel, 2416
  - setLoggerName, 2416
  - setMessage, 2416
  - setSourceFile, 2416
  - setSourceFunction, 2416
  - setSourceLine, 2416
  - setThrown, 2417
  - setTimestamp, 2417
  - setTreadId, 2417
- decaf::util::logging::LogWriter, 2418
  - ~LogWriter, 2418
  - destroy, 2418
  - getInstance, 2418
  - log, 2418, 2419
  - LogWriter, 2418
  - returnInstance, 2419
- decaf::util::logging::MarkBlockLogger, 2483
  - ~MarkBlockLogger, 2483
  - MarkBlockLogger, 2483
- decaf::util::logging::PropertiesChangeListener, 3135
  - ~PropertiesChangeListener, 3135
  - onPropertiesReset, 3135
  - onPropertyChanged, 3135
- decaf::util::logging::SimpleFormatter, 3500
  - ~SimpleFormatter, 3500
  - format, 3500
  - SimpleFormatter, 3500
- decaf::util::logging::SimpleLogger, 3501
  - ~SimpleLogger, 3501
  - debug, 3502
  - error, 3502
  - fatal, 3502
  - info, 3502
  - log, 3502
  - mark, 3502
  - SimpleLogger, 3501
  - warn, 3502
- decaf::util::logging::StreamHandler, 3649
  - ~StreamHandler, 3650
  - close, 3650

- flush, 3650
- isLoggable, 3651
- publish, 3651
- setOutputStream, 3651
- StreamHandler, 3650
- decaf::util::logging::XMLFormatter, 4060
  - ~XMLFormatter, 4060
  - format, 4060
  - getHead, 4061
  - getTail, 4061
  - XMLFormatter, 4060
- decaf::util::Map, 2459
  - ~Map, 2460
  - clear, 2460
  - containsKey, 2461
  - containsValue, 2462
  - copy, 2463
  - equals, 2463
  - get, 2463, 2464
  - isEmpty, 2465
  - keySet, 2466
  - Map, 2460
  - put, 2467
  - putAll, 2467
  - remove, 2468
  - size, 2469
  - values, 2470
- decaf::util::Map::Entry, 1824
  - ~Entry, 1824
  - Entry, 1824
  - getKey, 1824
  - getValue, 1824
  - setValue, 1824
- decaf::util::PriorityQueue, 3028
  - ~PriorityQueue, 3031
  - add, 3031
  - clear, 3031
  - comparator, 3031
  - iterator, 3032
  - offer, 3032
  - operator=, 3032, 3033
  - peek, 3033
  - poll, 3033
  - PriorityQueue, 3030, 3031
  - PriorityQueueIterator, 3035
  - remove, 3033, 3034
  - size, 3034
- decaf::util::Properties, 3126
  - ~Properties, 3128
  - clear, 3128
  - clone, 3128
  - copy, 3128
  - defaults, 3134
  - equals, 3128
  - getProperty, 3128, 3129
  - hasProperty, 3129
  - isEmpty, 3129
  - load, 3129, 3131
  - operator=, 3131
  - Properties, 3128
  - propertyNames, 3132
  - remove, 3132
  - setProperty, 3132
  - size, 3132
  - store, 3132, 3133
  - toArray, 3134
  - toString, 3134
- decaf::util::Queue, 3149
  - ~Queue, 3150
  - element, 3150
  - offer, 3150
  - peek, 3150
  - poll, 3151
  - remove, 3151
- decaf::util::Random, 3155
  - next, 3156
  - nextBoolean, 3157
  - nextBytes, 3157
  - nextDouble, 3157
  - nextFloat, 3158
  - nextGaussian, 3158
  - nextInt, 3158
  - nextLong, 3159
  - Random, 3156
  - setSeed, 3159
- decaf::util::Set, 3439
  - ~Set, 3439
- decaf::util::StlList, 3589
  - ~StlList, 3594
  - add, 3594
  - addAll, 3595
  - clear, 3596
  - contains, 3596
  - copy, 3596
  - equals, 3596
  - get, 3597
  - indexOf, 3597
  - isEmpty, 3597
  - iterator, 3598
  - lastIndexOf, 3598
  - listIterator, 3598, 3599
  - remove, 3599, 3600
  - set, 3600
  - size, 3601
  - StlList, 3593, 3594
- decaf::util::StlMap, 3602
  - ~StlMap, 3606
  - clear, 3606

- containsKey, 3606
- containsValue, 3607
- copy, 3607
- equals, 3607, 3608
- get, 3608
- isEmpty, 3609
- keySet, 3609
- lock, 3609
- notify, 3609
- notifyAll, 3610
- put, 3610
- putAll, 3610, 3611
- remove, 3611
- size, 3611
- StlMap, 3605, 3606
- tryLock, 3612
- unlock, 3612
- values, 3612
- wait, 3612, 3613
- decaf::util::StlQueue, 3615
  - ~StlQueue, 3617
  - back, 3617
  - clear, 3617
  - empty, 3617
  - enqueueFront, 3618
  - front, 3618
  - getSafeValue, 3618
  - iterator, 3618
  - lock, 3618
  - notify, 3619
  - notifyAll, 3619
  - pop, 3619
  - push, 3619
  - reverse, 3620
  - size, 3620
  - StlQueue, 3617
  - toArray, 3620
  - tryLock, 3620
  - unlock, 3620
  - wait, 3621, 3622
- decaf::util::StlSet, 3623
  - ~StlSet, 3625
  - add, 3625
  - clear, 3626
  - contains, 3626
  - copy, 3627
  - equals, 3627
  - isEmpty, 3627
  - iterator, 3627
  - remove, 3628
  - size, 3628
  - StlSet, 3625
- decaf::util::StringTokenizer, 3668
  - ~StringTokenizer, 3669
  - countTokens, 3669
  - hasMoreTokens, 3669
  - nextToken, 3669
  - reset, 3670
  - StringTokenizer, 3668
  - toArray, 3670
- decaf::util::Timer, 3790
  - ~Timer, 3792
  - cancel, 3792
  - purge, 3792
  - schedule, 3792–3796
  - scheduleAtFixedRate, 3797–3799
  - Timer, 3792
- decaf::util::TimerTask, 3801
  - ~TimerTask, 3802
  - cancel, 3802
  - decaf::internal::util::TimerTaskHeap, 3803
  - getWhen, 3802
  - isScheduled, 3802
  - scheduledExecutionTime, 3802
  - setScheduledTime, 3802
  - Timer, 3803
  - TimerImpl, 3803
  - TimerTask, 3802
- decaf::util::UUID, 3969
  - ~UUID, 3971
  - clockSequence, 3971
  - compareTo, 3971
  - equals, 3971
  - fromString, 3971
  - getLeastSignificantBits, 3972
  - getMostSignificantBits, 3972
  - nameUUIDFromBytes, 3972
  - node, 3972
  - operator<, 3973
  - operator==, 3973
  - randomUUID, 3973
  - timestamp, 3973
  - toString, 3974
  - UUID, 3970
  - variant, 3974
  - version, 3974
- decaf::util::zip, 177
- decaf::util::zip::Adler32, 722
  - ~Adler32, 723
  - Adler32, 723
  - getValue, 723
  - reset, 723
  - update, 723, 724
- decaf::util::zip::CheckedInputStream, 1137
  - ~CheckedInputStream, 1138
  - CheckedInputStream, 1138
  - doReadArrayBounded, 1138
  - doReadByte, 1138



- getChecksum, 1138
  - skip, 1138
- decaf::util::zip::CheckedOutputStream, 1140
  - ~CheckedOutputStream, 1141
  - CheckedOutputStream, 1140
  - doWriteArrayBounded, 1141
  - doWriteByte, 1141
  - getChecksum, 1141
- decaf::util::zip::Checksum, 1142
  - ~Checksum, 1142
  - getValue, 1142
  - reset, 1143
  - update, 1143, 1144
- decaf::util::zip::CRC32, 1524
  - ~CRC32, 1525
  - CRC32, 1525
  - getValue, 1525
  - reset, 1525
  - update, 1525, 1526
- decaf::util::zip::DataFormatException, 1555
  - ~DataFormatException, 1556
  - clone, 1557
  - DataFormatException, 1555, 1556
- decaf::util::zip::Deflater, 1706
  - ~Deflater, 1708
  - BEST\_COMPRESSION, 1714
  - BEST\_SPEED, 1714
  - DEFAULT\_COMPRESSION, 1714
  - DEFAULT\_STRATEGY, 1714
  - deflate, 1709
  - DEFLATED, 1714
  - Deflater, 1708
  - end, 1710
  - FILTERED, 1715
  - finish, 1710
  - finished, 1710
  - getAdler, 1710
  - getBytesRead, 1710
  - getBytesWritten, 1711
  - HUFFMAN\_ONLY, 1715
  - needsInput, 1711
  - NO\_COMPRESSION, 1715
  - reset, 1711
  - setDictionary, 1711, 1712
  - setInput, 1712, 1713
  - setLevel, 1713
  - setStrategy, 1714
- decaf::util::zip::DeflaterOutputStream, 1716
  - ~DeflaterOutputStream, 1718
  - buf, 1719
  - close, 1718
  - DEFAULT\_BUFFER\_SIZE, 1719
  - deflate, 1718
  - deflater, 1719
  - DeflaterOutputStream, 1717, 1718
  - doWriteArrayBounded, 1718
  - doWriteByte, 1719
  - finish, 1719
  - isDone, 1719
  - ownDeflater, 1719
- decaf::util::zip::Inflater, 2027
  - ~Inflater, 2029
  - end, 2029
  - finish, 2029
  - finished, 2029
  - getAdler, 2029
  - getBytesRead, 2029
  - getBytesWritten, 2030
  - getRemaining, 2030
  - inflate, 2030, 2031
  - Inflater, 2029
  - needsDictionary, 2031
  - needsInput, 2032
  - reset, 2032
  - setDictionary, 2032, 2033
  - setInput, 2033, 2034
- decaf::util::zip::InflaterInputStream, 2035
  - ~InflaterInputStream, 2038
  - atEOF, 2041
  - available, 2038
  - buff, 2041
  - close, 2039
  - DEFAULT\_BUFFER\_SIZE, 2041
  - doReadArrayBounded, 2039
  - doReadByte, 2039
  - fill, 2039
  - inflater, 2041
  - InflaterInputStream, 2037, 2038
  - length, 2041
  - mark, 2039
  - markSupported, 2040
  - ownInflater, 2041
  - reset, 2040
  - skip, 2040
- decaf::util::zip::ZipException, 4064
  - ~ZipException, 4065
  - clone, 4066
  - ZipException, 4064, 4065
- DECAF\_API
  - decaf/util/Config.h, 4229
- DECAF\_CATCH\_EXCEPTION\_CONVERT
  - decaf/lang/exceptions/ExceptionDe-  
fines.h, 4173
- DECAF\_CATCH\_NOTHROW
  - decaf/lang/exceptions/ExceptionDe-  
fines.h, 4173
- DECAF\_CATCH\_RETHROW

- decaf/lang/exceptions/ExceptionDefines.h, 4174
- DECAF\_CATCHALL\_NOTHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4174
- DECAF\_CATCHALL\_THROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4174
- DECAF\_UNUSED
  - decaf/util/Config.h, 4229
- DecafRuntime
  - decaf::internal::DecafRuntime, 1670
- decode
  - decaf::internal::net::URLEncoderDecoder, 3932
  - decaf::lang::Byte, 963
  - decaf::lang::Integer, 2081
  - decaf::lang::Long, 2424
  - decaf::lang::Short, 3443
  - decaf::net::URLDecoder, 3962
- decreaseUsage
  - activemq::util::MemoryUsage, 2513
  - activemq::util::Usage, 3964
- decrementAndGet
  - decaf::util::concurrent::atomic::AtomicInteger, 743
- DedicatedTaskRunner
  - activemq::threads::DedicatedTaskRunner, 1671
- DEF\_MEM\_LEVEL
  - zutil.h, 4745
- DEF\_WBITS
  - zutil.h, 4745
- DEFAULT\_BUFFER\_SIZE
  - decaf::util::zip::DeflaterOutputStream, 1719
  - decaf::util::zip::InflaterInputStream, 2041
- DEFAULT\_COMPRESSION
  - decaf::util::zip::Deflater, 1714
- DEFAULT\_DURABLE\_TOPIC\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1676
- DEFAULT\_MAX\_BLOCK\_SIZE
  - decaf::util::concurrent::ThreadPool, 3782
- DEFAULT\_MAX\_POOL\_SIZE
  - decaf::util::concurrent::ThreadPool, 3782
- DEFAULT\_MESSAGE\_SIZE
  - activemq::commands::Message, 2532
- DEFAULT\_ORDERED\_TARGET
  - activemq::commands::ActiveMQDestination, 331
- DEFAULT\_PRIORITY
  - activemq::cmsutil::CmsTemplate, 1182
- DEFAULT\_QUEUE\_BROWSER\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1676
- DEFAULT\_QUEUE\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1676
- DEFAULT\_STRATEGY
  - decaf::util::zip::Deflater, 1714
- DEFAULT\_TIME\_TO\_LIVE
  - activemq::cmsutil::CmsTemplate, 1182
- DEFAULT\_TOPIC\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1676
- DEFAULT\_URI
  - activemq::core::ActiveMQConnectionFactory, 302
- DEFAULT\_VERSION
  - activemq::wireformat::openwire::OpenWireFormat, 2898
- DefaultPrefetchPolicy
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
- DefaultRedeliveryPolicy
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
- defaults
  - decaf::util::Properties, 3134
- DefaultServerSocketFactory
  - decaf::internal::net::DefaultServerSocketFactory, 1683
- DefaultSocketFactory
  - decaf::internal::net::DefaultSocketFactory, 1688
- DefaultSSLContext
  - decaf::internal::net::ssl::DefaultSSLContext, 1691
- DefaultSSLServerSocketFactory
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1694
- DefaultSSLSocketFactory
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1699
- deflate
  - decaf::util::zip::Deflater, 1709
  - decaf::util::zip::DeflaterOutputStream, 1718
- deflate.h
  - \_dist\_code, 4727
  - \_length\_code, 4727
  - \_tr\_tally\_dist, 4726
  - \_tr\_tally\_lit, 4726
  - BL\_CODES, 4726
  - BUSY\_STATE, 4727

- Code, 4727
- COMMENT\_STATE, 4727
- ct\_data, 4727
- d\_code, 4727
- D\_CODES, 4727
- Dad, 4727
- deflate\_state, 4727
- EXTRA\_STATE, 4727
- FINISH\_STATE, 4727
- Freq, 4727
- GZIP, 4727
- HCRC\_STATE, 4727
- HEAP\_SIZE, 4727
- INIT\_STATE, 4727
- IPos, 4727
- L\_CODES, 4727
- Len, 4727
- LENGTH\_CODES, 4727
- LITERALS, 4727
- MAX\_BITS, 4727
- MAX\_DIST, 4727
- max\_insert\_length, 4727
- MIN\_LOOKAHEAD, 4727
- NAME\_STATE, 4727
- OF, 4727
- Pos, 4727
- Posf, 4727
- put\_byte, 4727
- static\_tree\_desc, 4727
- tree\_desc, 4727
- WIN\_INIT, 4727
- deflate\_state
  - deflate.h, 4727
- DEFLATED
  - decaf::util::zip::Deflater, 1714
- deflateInit
  - zlib.h, 4741
- deflateInit2
  - zlib.h, 4741
- Deflater
  - decaf::util::zip::Deflater, 1708
- deflater
  - decaf::util::zip::DeflaterOutputStream, 1719
- DeflaterOutputStream
  - decaf::util::zip::DeflaterOutputStream, 1717, 1718
- deleteIfCancelled
  - decaf::internal::util::TimerTaskHeap, 3805
- deliverAcks
  - activemq::core::ActiveMQConsumer, 314
  - activemq::core::ActiveMQSession, 526
- DELIVERY\_MODE
  - cms::DeliveryMode, 1721
- deliverySequenceId
  - activemq::commands::MessageDispatchNotification, 2634
- depth
  - internal\_state, 2120
- dequeue
  - activemq::core::ActiveMQConsumer, 314
  - activemq::core::MessageDispatchChannel, 2600
- dequeueNoWait
  - activemq::core::MessageDispatchChannel, 2601
- deQueueTask
  - decaf::util::concurrent::ThreadPool, 3779
- descriptor
  - decaf::io::FileDescriptor, 1892
- destination
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3066
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3176
  - activemq::commands::ConsumerControl, 1405
  - activemq::commands::ConsumerInfo, 1466
  - activemq::commands::DestinationInfo, 1731
  - activemq::commands::JournalQueueAck, 2158
  - activemq::commands::JournalTopicAck, 2187
  - activemq::commands::Message, 2532
  - activemq::commands::MessageAck, 2564
  - activemq::commands::MessageDispatch, 2598
  - activemq::commands::MessageDispatchNotification, 2634
  - activemq::commands::MessagePull, 2743
  - activemq::commands::ProducerInfo, 3100
  - activemq::commands::SubscriptionInfo, 3675
- DESTINATION\_ADD\_OPERATION
  - activemq::core::ActiveMQConstants, 308
- DESTINATION\_REMOVE\_OPERATION
  - activemq::core::ActiveMQConstants, 308
- DestinationActions
  - activemq::core::ActiveMQConstants, 308
- DestinationInfo
  - activemq::commands::DestinationInfo, 1728
- DestinationInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::DestinationInfo, 1745
  - activemq::wireformat::openwire::marshal::v2::DestinationInfo, 1733

- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1102
- 1737
- direct
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1741
- DISCONNECT
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller; stomp::StompCommandConstants, 1753
- 3631
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1749
- activemq::commands::DiscoveryEvent, 1759
- DiscoveryEventMarshaller
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1779
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1771
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1775
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1775
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1783
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1763
- dispatch
- activemq::core::ActiveMQConsumer, 314
- activemq::core::ActiveMQSession, 526
- activemq::core::Dispatcher, 1787
- dispatchAsync
- activemq::commands::ConsumerInfo, 1466
- activemq::commands::ProducerInfo, 3100
- activemq::commands::ActiveMQTempQueue, DispatchData
- activemq::core::DispatchData, 1786
- activemq::commands::ActiveMQTempTopic, DIST
- inflate.h, 4733
- distbits
- inflate\_state, 2025
- listcode
- inflate\_state, 2025
- DISTEXT
- inflate.h, 4733
- DISTS
- inftrees.h, 4734
- dl
- ct\_data\_s, 1527
- dmax
- inflate\_state, 2025
- doAppendChar
- decaf::io::Writer, 4023
- doAppendCharSequence
- decaf::io::Writer, 4024
- doAppendCharSequenceStartEnd
- decaf::io::Writer, 4024
- doClose
- activemq::core::ActiveMQConsumer, 314
- doCreateComposite
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1102
- 1737
- direct
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1741
- DISCONNECT
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller; stomp::StompCommandConstants, 1753
- 3631
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1749
- activemq::commands::DiscoveryEvent, 1759
- DiscoveryEventMarshaller
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1779
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1771
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1775
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1775
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1783
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1763
- dispatch
- activemq::core::ActiveMQConsumer, 314
- activemq::core::ActiveMQSession, 526
- activemq::core::Dispatcher, 1787
- dispatchAsync
- activemq::commands::ConsumerInfo, 1466
- activemq::commands::ProducerInfo, 3100
- activemq::commands::ActiveMQTempQueue, DispatchData
- activemq::core::DispatchData, 1786
- activemq::commands::ActiveMQTempTopic, DIST
- inflate.h, 4733
- distbits
- inflate\_state, 2025
- listcode
- inflate\_state, 2025
- DISTEXT
- inflate.h, 4733
- DISTS
- inftrees.h, 4734
- dl
- ct\_data\_s, 1527
- dmax
- inflate\_state, 2025
- doAppendChar
- decaf::io::Writer, 4023
- doAppendCharSequence
- decaf::io::Writer, 4024
- doAppendCharSequenceStartEnd
- decaf::io::Writer, 4024
- doClose
- activemq::core::ActiveMQConsumer, 314
- doCreateComposite
- DestinationOption
- activemq::core::ActiveMQConstants, 308
- DestinationType
- cms::Destination, 1723
- destOptionMap
- activemq::core::ActiveMQConstants::StaticInitializer, 1767
- 3588
- destOptions
- activemq::core::ActiveMQConstants::StaticInitializer, 1767
- 3588
- destroy
- activemq::cmsutil::CmsAccessor, 1155
- activemq::cmsutil::CmsDestinationAccessor, 1158
- activemq::cmsutil::CmsTemplate, 1173
- activemq::cmsutil::DestinationResolver, 1756
- activemq::cmsutil::DynamicDestinationResolver, 1822
- activemq::cmsutil::ResourceLifecycleManager, 3284
- activemq::commands::ActiveMQTempQueue, DispatchData
- 606
- activemq::commands::ActiveMQTempTopic, DIST
- 635
- cms::TemporaryQueue, 3759
- cms::TemporaryTopic, 3761
- decaf::internal::util::concurrent::ConditionImpl, 1257
- decaf::internal::util::concurrent::MutexImpl, 2788
- decaf::util::logging::LogWriter, 2418
- destroyDestination
- activemq::core::ActiveMQConnection, 279, 280
- destroyMarshalers
- activemq::wireformat::openwire::OpenWireFormat, 2890
- destroyResources
- decaf::internal::util::ResourceLifecycleManager, 3281
- DICT
- inflate.h, 4732
- DICTID
- inflate.h, 4732
- digit

- activemq::transport::failover::FailoverTransportFactory, 1886
- activemq::transport::mock::MockTransportFactory, 2780
- activemq::transport::tcp::SslTransportFactory, 3576
- activemq::transport::tcp::TcpTransportFactory, 3757
- doInCms
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3065
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3350
  - activemq::cmsutil::ProducerCallback, 3064
  - activemq::cmsutil::SessionCallback, 3378
- DONE
  - inflate.h, 4733
- done
  - gz\_header\_s, 1975
- doReadArray
  - decaf::io::FilterInputStream, 1897
  - decaf::io::InputStream, 2045
  - decaf::io::Reader, 3164
- doReadArrayBounded
  - activemq::io::LoggingInputStream, 2399
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
  - decaf::internal::net::tcp::TcpSocketInputStream, 3748
  - decaf::io::BlockingByteArrayInputStream, 841
  - decaf::io::BufferedInputStream, 937
  - decaf::io::ByteArrayInputStream, 1024
  - decaf::io::FilterInputStream, 1897
  - decaf::io::InputStream, 2045
  - decaf::io::InputStreamReader, 2054
  - decaf::io::PushbackInputStream, 3144
  - decaf::io::Reader, 3164
  - decaf::util::zip::CheckedInputStream, 1138
  - decaf::util::zip::InflaterInputStream, 2039
- doReadByte
  - activemq::io::LoggingInputStream, 2399
  - decaf::internal::io::StandardInputStream, 3582
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
  - decaf::internal::net::tcp::TcpSocketInputStream, 3749
  - decaf::io::BlockingByteArrayInputStream, 841
  - decaf::io::BufferedInputStream, 937
  - decaf::io::ByteArrayInputStream, 1024
- decaf::io::FilterInputStream, 1897
- decaf::io::InputStream, 2045
- decaf::io::PushbackInputStream, 3144
- decaf::util::zip::CheckedInputStream, 1138
- decaf::util::zip::InflaterInputStream, 2039
- doReadChar
  - decaf::io::Reader, 3164
- doReadCharBuffer
  - decaf::io::Reader, 3165
- doReadVector
  - decaf::io::Reader, 3165
- doStartTransaction
  - activemq::core::ActiveMQSession, 526
- Double
  - decaf::lang::Double, 1790
- DOUBLE\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- DoubleArrayBuffer
  - decaf::internal::nio::DoubleArrayBuffer, 1802, 1803
- DoubleBuffer
  - decaf::nio::DoubleBuffer, 1811
- doubleToLongBits
  - decaf::lang::Double, 1791
- doubleToRawLongBits
  - decaf::lang::Double, 1792
- doubleValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
  - decaf::lang::Byte, 963
  - decaf::lang::Character, 1103
  - decaf::lang::Double, 1792
  - decaf::lang::Float, 1907
  - decaf::lang::Integer, 2081
  - decaf::lang::Long, 2424
  - decaf::lang::Number, 2836
  - decaf::lang::Short, 3443
  - decaf::util::concurrent::atomic::AtomicInteger, 743
- doUnmarshal
  - activemq::wireformat::openwire::OpenWireFormat, 2890
- doWriteArray
  - decaf::io::BufferedOutputStream, 941
  - decaf::io::FilterOutputStream, 1901
  - decaf::io::OutputStream, 2909
  - decaf::io::Writer, 4024
- doWriteArrayBounded
  - activemq::io::LoggingOutputStream, 2401
  - decaf::internal::io::StandardErrorOutputStream, 3580
  - decaf::internal::io::StandardOutputStream, 3585

- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2886
- decaf::internal::net::tcp::TcpSocketOutputStream, 3751
- decaf::io::BufferedOutputStream, 941
- decaf::io::ByteArrayOutputStream, 1029
- decaf::io::DataOutputStream, 1580
- decaf::io::FilterOutputStream, 1902
- decaf::io::OutputStream, 2909
- decaf::io::OutputStreamWriter, 2916
- decaf::io::Writer, 4024
- decaf::util::zip::CheckedOutputStream, 1141
- decaf::util::zip::DeflaterOutputStream, 1718
- doWriteByte
  - activemq::io::LoggingOutputStream, 2401
- decaf::internal::io::StandardErrorOutputStream, 3580
- decaf::internal::io::StandardOutputStream, 3585
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2886
- decaf::internal::net::tcp::TcpSocketOutputStream, 3751
- decaf::io::BufferedOutputStream, 941
- decaf::io::ByteArrayOutputStream, 1029
- decaf::io::DataOutputStream, 1580
- decaf::io::FilterOutputStream, 1902
- decaf::io::OutputStream, 2909
- decaf::util::zip::CheckedOutputStream, 1141
- decaf::util::zip::DeflaterOutputStream, 1719
- doWriteChar
  - decaf::io::Writer, 4024
- doWriteString
  - decaf::io::Writer, 4024
- doWriteStringBounded
  - decaf::io::Writer, 4024
- doWriteVector
  - decaf::io::Writer, 4024
- drainPermits
  - decaf::util::concurrent::Semaphore, 3346
- drainTo
  - decaf::util::concurrent::BlockingQueue, 846
  - decaf::util::concurrent::SynchronousQueue, 3719, 3720
- droppable
  - activemq::commands::Message, 2532
- dummy
  - internal\_state, 2120
- duplexConnection
  - activemq::commands::BrokerInfo, 903
- decaf::internal::nio::ByteBuffer, 1007
- decaf::internal::nio::CharArrayBuffer, 1114
- decaf::internal::nio::DoubleArrayBuffer, 1805
- decaf::internal::nio::FloatArrayBuffer, 1921
- decaf::internal::nio::IntArrayBuffer, 2062
- decaf::internal::nio::LongArrayBuffer, 2440
- decaf::internal::nio::ShortArrayBuffer, 3455
- decaf::nio::ByteBuffer, 1040
- decaf::nio::CharBuffer, 1126
- decaf::nio::DoubleBuffer, 1813
- decaf::nio::FloatBuffer, 1929
- decaf::nio::IntBuffer, 2070
- decaf::nio::LongBuffer, 2448
- decaf::nio::ShortBuffer, 3463
- DUPS\_OK\_ACKNOWLEDGE
  - cms::Session, 3368
- dyn\_dtree
  - internal\_state, 2120
- dyn\_output
  - internal\_state, 2120
- dyn\_tree
  - tree\_desc\_s, 3905
- DYN\_TREES
  - zutil.h, 4745
- dynamicCast
  - decaf::lang::Pointer, 2950
- DynamicDestinationResolver
  - activemq::cmsutil::DynamicDestinationResolver, 1822
- E
  - decaf::lang::Math, 2511
- element
  - decaf::util::AbstractQueue, 196
  - decaf::util::Queue, 3150
- empty
  - decaf::util::StlQueue, 3617
- encode
  - decaf::net::URLEncoder, 3963
- encodeOthers
  - decaf::internal::net::URLEncoderDecoder, 3933
- end
  - decaf::util::zip::Deflater, 1710
  - decaf::util::zip::Inflater, 2029
- ENOUGH
  - inftrees.h, 4734
- ENOUGH\_DISTS
  - inftrees.h, 4734
- ENOUGH\_LENS
  - inftrees.h, 4734

- enqueue
  - activemq::core::MessageDispatchChannel, 2601
- enqueueFirst
  - activemq::core::MessageDispatchChannel, 2601
- enqueueFront
  - decaf::util::StlQueue, 3618
- enqueueUsage
  - activemq::util::MemoryUsage, 2513
  - activemq::util::Usage, 3964
- ensureCreated
  - decaf::net::ServerSocket, 3357
  - decaf::net::Socket, 3510
- entering
  - decaf::util::logging::Logger, 2390
- Entry
  - decaf::util::Map::Entry, 1824
- eof
  - gz\_state, 1977
- EOFException
  - decaf::io::EOFException, 1825, 1826
- equals
  - activemq::commands::ActiveMQBlobMessage, 206
  - activemq::commands::ActiveMQBytesMessage, 237
  - activemq::commands::ActiveMQDestination, 325
  - activemq::commands::ActiveMQMapMessage, 364
  - activemq::commands::ActiveMQMessage, 399
  - activemq::commands::ActiveMQMessageTemplate, 429
  - activemq::commands::ActiveMQObjectMessage, 444
  - activemq::commands::ActiveMQQueue, 484
  - activemq::commands::ActiveMQStreamMessage, 541
  - activemq::commands::ActiveMQTempDestination, 577
  - activemq::commands::ActiveMQTempQueue, 606
  - activemq::commands::ActiveMQTempTopic, 635
  - activemq::commands::ActiveMQTextMessage, 664
  - activemq::commands::ActiveMQTopic, 692
  - activemq::commands::BaseCommand, 759
  - activemq::commands::BaseDataStructure, 832
  - activemq::commands::BooleanExpression, 856
  - activemq::commands::BrokerId, 870
  - activemq::commands::BrokerInfo, 898
  - activemq::commands::ConnectionControl, 1268
  - activemq::commands::ConnectionError, 1297
  - activemq::commands::ConnectionId, 1328, 1329
  - activemq::commands::ConnectionInfo, 1357
  - activemq::commands::ConsumerControl, 1402
  - activemq::commands::ConsumerId, 1431, 1432
  - activemq::commands::ConsumerInfo, 1461
  - activemq::commands::ControlCommand, 1494
  - activemq::commands::DataArrayResponse, 1529
  - activemq::commands::DataResponse, 1584
  - activemq::commands::DataStructure, 1661
  - activemq::commands::DestinationInfo, 1728
  - activemq::commands::DiscoveryEvent, 1759
  - activemq::commands::ExceptionResponse, 1840
  - activemq::commands::FlushCommand, 1938
  - activemq::commands::IntegerResponse, 2092
  - activemq::commands::JournalQueueAck, 2157
  - activemq::commands::JournalTopicAck, 2184
  - activemq::commands::JournalTrace, 2213
  - activemq::commands::JournalTransaction, 2240
  - activemq::commands::KeepAliveInfo, 2267
  - activemq::commands::LastPartialCommand, 2302
  - activemq::commands::LocalTransactionId, 2348, 2349
  - activemq::commands::Message, 2521
  - activemq::commands::MessageAck, 2560
  - activemq::commands::MessageDispatch, 2595
  - activemq::commands::MessageDispatchNotification, 2631
  - activemq::commands::MessageId, 2665
  - activemq::commands::MessagePull, 2740

- activemq::commands::NetworkBridgeFilter, 2794
- activemq::commands::PartialCommand, 2919
- activemq::commands::ProducerAck, 3037
- activemq::commands::ProducerId, 3069
- activemq::commands::ProducerInfo, 3097
- activemq::commands::RemoveInfo, 3195
- activemq::commands::RemoveSubscriptionInfo, 3223
- activemq::commands::ReplayCommand, 3252
- activemq::commands::Response, 3286
- activemq::commands::SessionId, 3381
- activemq::commands::SessionInfo, 3408
- activemq::commands::ShutdownInfo, 3471
- activemq::commands::SubscriptionInfo, 3672
- activemq::commands::TransactionId, 3820
- activemq::commands::TransactionInfo, 3848
- activemq::commands::WireFormatInfo, 3985
- activemq::commands::XATransactionId, 4033
- decaf::lang::Boolean, 852
- decaf::lang::Byte, 963
- decaf::lang::Character, 1103
- decaf::lang::Comparable, 1215
- decaf::lang::Double, 1792, 1793
- decaf::lang::Float, 1907, 1908
- decaf::lang::Integer, 2081, 2082
- decaf::lang::Long, 2424, 2425
- decaf::lang::Short, 3443
- decaf::net::URI, 3925
- decaf::nio::ByteBuffer, 1040
- decaf::nio::CharBuffer, 1126
- decaf::nio::DoubleBuffer, 1814
- decaf::nio::FloatBuffer, 1929
- decaf::nio::IntBuffer, 2070
- decaf::nio::LongBuffer, 2448
- decaf::nio::ShortBuffer, 3463
- decaf::security::cert::Certificate, 1087
- decaf::security::Principal, 3026
- decaf::util::AbstractCollection, 185
- decaf::util::Collection, 1189
- decaf::util::concurrent::ConcurrentStlMap, 1239
- decaf::util::concurrent::SynchronousQueue, 3720
- decaf::util::concurrent::TimeUnit, 3810
- decaf::util::Date, 1667
- decaf::util::logging::Level, 2334
- decaf::util::Map, 2463
- decaf::util::Properties, 3128
- decaf::util::StlList, 3596
- decaf::util::StlMap, 3607, 3608
- decaf::util::StlSet, 3627
- decaf::util::UUID, 3971
- err
  - decaf::io::FileDescriptor, 1892
  - gz\_state, 1977
- ERR\_MSG
  - zutil.h, 4745
- ERR\_RETURN
  - zutil.h, 4745
- Error
  - decaf::util::logging, 176
- error
  - decaf::util::logging::ErrorManager, 1829
  - decaf::util::logging::SimpleLogger, 3502
- ERROR\_CMD
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- ErrorManager
  - decaf::util::logging::ErrorManager, 1829
- Exception
  - decaf::lang::Exception, 1832, 1833
- exception
  - activemq::commands::ConnectionError, 1299
  - activemq::commands::ExceptionResponse, 1841
- ExceptionResponse
  - activemq::commands::ExceptionResponse, 1840
- ExceptionResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ExceptionResponse, 1863
  - activemq::wireformat::openwire::marshal::v2::ExceptionResponse, 1847
  - activemq::wireformat::openwire::marshal::v3::ExceptionResponse, 1851
  - activemq::wireformat::openwire::marshal::v4::ExceptionResponse, 1859
  - activemq::wireformat::openwire::marshal::v5::ExceptionResponse, 1855
  - activemq::wireformat::openwire::marshal::v6::ExceptionResponse, 1843
- exclusive
  - activemq::commands::ActiveMQDestination, 331
  - activemq::commands::ConsumerInfo, 1466
- execute
  - activemq::cmsutil::CmsTemplate, 1174
  - activemq::core::ActiveMQSessionExecutor, 534
  - decaf::util::concurrent::Executor, 1870



- executeFirst
  - activemq::core::ActiveMQSessionExecutor, 534
- ExecutionException
  - decaf::util::concurrent::ExecutionException, 1866, 1867
- exit
  - activemq::commands::ConnectionControl, 1271
- exiting
  - decaf::util::logging::Logger, 2390
- EXLEN
  - inflate.h, 4732
- expiration
  - activemq::commands::Message, 2532
- EXTRA
  - inflate.h, 4732
- extra
  - gz\_header\_s, 1975
  - inflate\_state, 2025
- extra\_len
  - gz\_header\_s, 1975
- extra\_max
  - gz\_header\_s, 1975
- EXTRA\_STATE
  - deflate.h, 4727
- F\_OPEN
  - zutil.h, 4745
- failIfReadOnlyBody
  - activemq::commands::ActiveMQMessageTemplate, 429
- failIfReadOnlyProperties
  - activemq::commands::ActiveMQMessageTemplate, 429
- failIfWriteOnlyBody
  - activemq::commands::ActiveMQMessageTemplate, 429
- FailoverTransport
  - activemq::transport::failover::FailoverTransport, 1875
- FailoverTransportListener
  - activemq::transport::failover::FailoverTransport, 1884
  - activemq::transport::failover::FailoverTransportListener, 1889
- Fatal
  - decaf::util::logging, 176
- fatal
  - decaf::util::logging::SimpleLogger, 3502
- faultTolerant
  - activemq::commands::ConnectionControl, 1271
- activemq::commands::ConnectionInfo, 1360
- faultTolerantConfiguration
  - activemq::commands::BrokerInfo, 903
- fc
  - ct\_data\_s, 1527
- fd
  - decaf::net::SocketImpl, 3536
  - gz\_state, 1977
- FileDescriptor
  - decaf::io::FileDescriptor, 1892
- FileName
  - activemq::commands::BrokerError::StackTraceElement, 3578
- fill
  - decaf::util::zip::InflaterInputStream, 2039
- FILTERED
  - decaf::util::zip::Deflater, 1715
- FilterInputStream
  - decaf::io::FilterInputStream, 1896
- FilterOutputStream
  - decaf::io::FilterOutputStream, 1901
- find
  - decaf::internal::util::TimerTaskHeap, 3805
- findFactory
  - activemq::transport::TransportRegistry, 3903
  - activemq::wireformat::WireFormatRegistry, 4018
- FINE
  - decaf::util::logging::Level, 2335
  - decaf::util::logging::Logger, 2390
- FINER
  - decaf::util::logging::Level, 2335
- finer
  - decaf::util::logging::Logger, 2391
- FINEST
  - decaf::util::logging::Level, 2335
- finest
  - decaf::util::logging::Logger, 2391
- finish
  - decaf::util::zip::Deflater, 1710
  - decaf::util::zip::DeflaterOutputStream, 1719
  - decaf::util::zip::Inflater, 2029
- FINISH\_STATE
  - deflate.h, 4727
- finished
  - decaf::util::zip::Deflater, 1710
  - decaf::util::zip::Inflater, 2029
- fire
  - activemq::core::ActiveMQConnection, 280
  - activemq::core::ActiveMQSession, 526

- activemq::transport::TransportFilter, 3893
- fireCommand
  - activemq::transport::mock::MockTransport, 2770
- fireException
  - activemq::transport::mock::MockTransport, 2771
- firstMessageId
  - activemq::commands::MessageAck, 2564
- firstNakNumber
  - activemq::commands::ReplayCommand, 3253
- FLAGS
  - inflate.h, 4732
- flags
  - inflate\_state, 2025
- flip
  - decaf::nio::Buffer, 931
- Float
  - decaf::lang::Float, 1906
- FLOAT\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- FloatArrayBuffer
  - decaf::internal::nio::FloatArrayBuffer, 1918, 1919
- FloatBuffer
  - decaf::nio::FloatBuffer, 1927
- floatToIntBits
  - decaf::lang::Float, 1908
- floatToRawIntBits
  - decaf::lang::Float, 1908
- floatValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
  - decaf::lang::Byte, 963
  - decaf::lang::Character, 1103
  - decaf::lang::Double, 1793
  - decaf::lang::Float, 1909
  - decaf::lang::Integer, 2082
  - decaf::lang::Long, 2425
  - decaf::lang::Number, 2836
  - decaf::lang::Short, 3443
  - decaf::util::concurrent::atomic::AtomicInteger, 743
- floor
  - decaf::lang::Math, 2501
- flush
  - activemq::commands::ConsumerControl, 1405
  - decaf::internal::io::StandardErrorOutputStream, 3580
  - decaf::internal::io::StandardOutputStream, 3585
  - decaf::io::BufferedOutputStream, 941
  - decaf::io::FilterOutputStream, 1902
  - decaf::io::Flushable, 1936
  - decaf::io::OutputStream, 2910
  - decaf::io::OutputStreamWriter, 2916
  - decaf::util::logging::Handler, 1979
  - decaf::util::logging::StreamHandler, 3650
- FLUSH\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- FlushCommand
  - activemq::commands::FlushCommand, 1938
- FlushCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::FlushCommand, 1957
  - activemq::wireformat::openwire::marshal::v2::FlushCommand, 1945
  - activemq::wireformat::openwire::marshal::v3::FlushCommand, 1949
  - activemq::wireformat::openwire::marshal::v4::FlushCommand, 1953
  - activemq::wireformat::openwire::marshal::v5::FlushCommand, 1961
  - activemq::wireformat::openwire::marshal::v6::FlushCommand, 1941
- format
  - decaf::util::logging::Formatter, 1964
  - decaf::util::logging::SimpleFormatter, 3500
  - decaf::util::logging::XMLFormatter, 4060
- FORMAT\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- formatId
  - activemq::commands::XATransactionId, 4035
- formatMessage
  - decaf::util::logging::Formatter, 1965
- Freq
  - deflate.h, 4727
- freq
  - ct\_data\_s, 1527
- fromStream
  - activemq::wireformat::stomp::StompFrame, 3634
- fromString
  - decaf::util::UUID, 3971
- front
  - decaf::util::StlQueue, 3618
- FutureResponse
  - activemq::transport::correlator::FutureResponse, 1969
- GeneralSecurityException
  - decaf::security::GeneralSecurityException, 1971, 1972
- generateId

- activemq::util::IdGenerator, 1990
- generation
  - decaf::util::concurrent::ConditionHandle, 1255
- GENERIC\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- GenericResource
  - decaf::internal::util::GenericResource, 1974
- get
  - decaf::internal::nio::ByteBuffer, 1007, 1008
  - decaf::internal::nio::CharArrayBuffer, 1115
  - decaf::internal::nio::DoubleArrayBuffer, 1805, 1806
  - decaf::internal::nio::FloatArrayBuffer, 1921, 1922
  - decaf::internal::nio::IntArrayBuffer, 2062, 2063
  - decaf::internal::nio::LongArrayBuffer, 2440, 2441
  - decaf::internal::nio::ShortArrayBuffer, 3455, 3456
  - decaf::internal::util::ByteArrayAdapter, 976
  - decaf::lang::ArrayPointer, 733
  - decaf::lang::Pointer, 2950
  - decaf::nio::ByteBuffer, 1040–1042
  - decaf::nio::CharBuffer, 1126, 1127
  - decaf::nio::DoubleBuffer, 1814, 1815
  - decaf::nio::FloatBuffer, 1929–1931
  - decaf::nio::IntBuffer, 2070–2072
  - decaf::nio::LongBuffer, 2448–2450
  - decaf::nio::ShortBuffer, 3463–3465
  - decaf::util::concurrent::atomic::AtomicBoolean, 739
  - decaf::util::concurrent::atomic::AtomicInteger, 743
  - decaf::util::concurrent::atomic::AtomicReference, 750
  - decaf::util::concurrent::ConcurrentStlMap, 1239, 1240
  - decaf::util::concurrent::Future, 1967
  - decaf::util::List, 2339
  - decaf::util::Map, 2463, 2464
  - decaf::util::StlList, 3597
  - decaf::util::StlMap, 3608
- getAckHandler
  - activemq::commands::Message, 2521
- getAckMode
  - activemq::commands::SessionInfo, 3408
- getAcknowledgeMode
  - activemq::cmsutil::PooledSession, 2965
  - activemq::core::ActiveMQSession, 526
  - cms::Session, 3375
- getAckType
  - activemq::commands::MessageAck, 2561
- getAdditionalPredicate
  - activemq::commands::ConsumerInfo, 1462
- getAddress
  - decaf::net::InetAddress, 2018
- getAdler
  - decaf::util::zip::Deflater, 1710
  - decaf::util::zip::Inflater, 2029
- getAlgorithm
  - decaf::security::Key, 2294
- getAndAdd
  - decaf::util::concurrent::atomic::AtomicInteger, 744
- getAndDecrement
  - decaf::util::concurrent::atomic::AtomicInteger, 744
- getAndIncrement
  - decaf::util::concurrent::atomic::AtomicInteger, 744
- getAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 739
  - decaf::util::concurrent::atomic::AtomicInteger, 744
  - decaf::util::concurrent::atomic::AtomicReference, 750
- getAnonymousLogger
  - decaf::util::logging::Logger, 2391
- getAnyAddress
  - decaf::net::InetAddress, 2018
- getAprPool
  - decaf::internal::AprPool, 728
- getArrival
  - activemq::commands::Message, 2522
- getAuthority
  - decaf::internal::net::URIType, 3954
- getBacklog
  - decaf::util::concurrent::ThreadPool, 3779
- getBackOffMultiplier
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
  - activemq::core::RedeliveryPolicy, 3178
  - activemq::transport::failover::FailoverTransport, 1876
- getBackup
  - activemq::transport::failover::BackupTransportPool, 756
- getBackupPoolSize
  - activemq::transport::failover::BackupTransportPool, 756
  - activemq::transport::failover::FailoverTransport, 1877

- getBasicConstraints
  - decaf::security::cert::X509Certificate, 4029
- getBlockSize
  - decaf::util::concurrent::ThreadPool, 3779
- getBody
  - activemq::wireformat::stomp::StompFrame, 3635
- getBodyBytes
  - activemq::commands::ActiveMQBytesMessage, 237
  - cms::BytesMessage, 1059
- getBodyLength
  - activemq::commands::ActiveMQBytesMessage, 237
  - activemq::wireformat::stomp::StompFrame, 3635
  - cms::BytesMessage, 1060
- getBool
  - activemq::util::PrimitiveList, 2983
  - activemq::util::PrimitiveMap, 2994
  - activemq::util::PrimitiveValueNode, 3018
- getBoolean
  - activemq::commands::ActiveMQMapMessage, 364
  - cms::MapMessage, 2474
- getBooleanProperty
  - activemq::commands::ActiveMQMessageTemplate, 429
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2734
  - cms::Message, 2540
- getBranchQualifier
  - activemq::commands::XATransactionId, 4033
- getBrokerId
  - activemq::commands::BrokerInfo, 898, 899
- getBrokerInTime
  - activemq::commands::Message, 2522
- getBrokerName
  - activemq::commands::BrokerInfo, 899
  - activemq::commands::DiscoveryEvent, 1759, 1760
- getBrokerOutTime
  - activemq::commands::Message, 2522
- getBrokerPath
  - activemq::commands::ConnectionInfo, 1357
  - activemq::commands::ConsumerInfo, 1462
  - activemq::commands::DestinationInfo, 1729
  - activemq::commands::Message, 2522
  - activemq::commands::ProducerInfo, 3098
- getBrokerSequenceId
  - activemq::commands::MessageId, 2665
- getBrokerUploadUrl
  - activemq::commands::BrokerInfo, 899
- getBrokerURL
  - activemq::commands::BrokerInfo, 899
  - activemq::core::ActiveMQConnection, 280
  - activemq::core::ActiveMQConnectionFactory, 296
- getByAddress
  - decaf::net::InetAddress, 2019
- getByte
  - activemq::commands::ActiveMQMapMessage, 364
  - activemq::util::PrimitiveList, 2983
  - activemq::util::PrimitiveMap, 2994
  - activemq::util::PrimitiveValueNode, 3019
  - cms::MapMessage, 2474
- getByteArray
  - activemq::util::PrimitiveList, 2983
  - activemq::util::PrimitiveMap, 2994
  - activemq::util::PrimitiveValueNode, 3019
  - decaf::internal::util::ByteArrayAdapter, 976
- getByteProperty
  - activemq::commands::ActiveMQMessageTemplate, 430
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2734
  - cms::Message, 2540
- getBytePropertyInterceptor
  - activemq::commands::ActiveMQMapMessage, 365
  - cms::MapMessage, 2475
- getBytesRead
  - decaf::util::zip::Deflater, 1710
  - decaf::util::zip::Inflater, 2029
- getBytesWritten
  - decaf::util::zip::Deflater, 1711
  - decaf::util::zip::Inflater, 2030
- getCacheSize
  - activemq::commands::WireFormatInfo, 3985
  - activemq::wireformat::openwire::OpenWireFormat, 2891
- getCapacity
  - decaf::internal::util::ByteArrayAdapter, 977
- getCause
  - activemq::commands::BrokerError, 864
  - cms::CMSException, 1161
  - decaf::lang::Exception, 1835
  - decaf::lang::Throwable, 3785
- getChar
  - activemq::commands::ActiveMQMapMessage, 365

- activemq::util::PrimitiveList, 2984
- activemq::util::PrimitiveMap, 2995
- activemq::util::PrimitiveValueNode, 3019
- cms::MapMessage, 2475
- decaf::internal::nio::ByteBuffer, 1008
- decaf::internal::util::ByteArrayAdapter, 977
- decaf::nio::ByteBuffer, 1042
- getCharArray
  - decaf::internal::util::ByteArrayAdapter, 977
- getCharCapacity
  - decaf::internal::util::ByteArrayAdapter, 977
- getChecksum
  - decaf::util::zip::CheckedInputStream, 1138
  - decaf::util::zip::CheckedOutputStream, 1141
- getCipherSuites
  - decaf::net::ssl::SSLParameters, 3552
- getClientID
  - activemq::core::ActiveMQConnection, 280
  - cms::Connection, 1264
- getClientId
  - activemq::commands::ActiveMQDestination, 326
  - activemq::commands::ConnectionInfo, 1357
  - activemq::commands::JournalTopicAck, 2184, 2185
  - activemq::commands::RemoveSubscriptionInfo, 3223, 3224
  - activemq::commands::SubscriptionInfo, 3672, 3673
  - activemq::core::ActiveMQConnectionFactory, 296
- getCloseTimeout
  - activemq::core::ActiveMQConnection, 281
  - activemq::core::ActiveMQConnectionFactory, 297
- getCluster
  - activemq::commands::Message, 2522
- getCMSCorrelationID
  - activemq::commands::ActiveMQMessageTemplate, 430
  - cms::Message, 2540
- getCMSDeliveryMode
  - activemq::commands::ActiveMQMessageTemplate, 430
  - cms::Message, 2541
- getCMSDestination
  - activemq::commands::ActiveMQDestination, 326
- activemq::commands::ActiveMQMessageTemplate, 431
- activemq::commands::ActiveMQQueue, 485
- activemq::commands::ActiveMQTempQueue, 606
- activemq::commands::ActiveMQTempTopic, 635
- activemq::commands::ActiveMQTopic, 693
- cms::Message, 2541
- getCMSExpiration
  - activemq::commands::ActiveMQMessageTemplate, 431
  - cms::Message, 2542
- getCMSMajorVersion
  - activemq::core::ActiveMQConnectionMetaData, 304
  - cms::ConnectionMetaData, 1386
- getCMSMessageID
  - activemq::commands::ActiveMQMessageTemplate, 431
  - cms::Message, 2542
- getCMSMinorVersion
  - activemq::core::ActiveMQConnectionMetaData, 304
  - cms::ConnectionMetaData, 1386
- getCMSPriority
  - activemq::commands::ActiveMQMessageTemplate, 431
  - cms::Message, 2543
- getCMSProperties
  - activemq::commands::ActiveMQQueue, 485
  - activemq::commands::ActiveMQTempQueue, 606
  - activemq::commands::ActiveMQTempTopic, 635
  - activemq::commands::ActiveMQTopic, 693
  - cms::Destination, 1724
- getCMSProviderName
  - activemq::core::ActiveMQConnectionMetaData, 304
  - cms::ConnectionMetaData, 1386
- getCMSRedelivered
  - activemq::commands::ActiveMQMessageTemplate, 432
  - cms::Message, 2543
- getCMSReplyTo
  - activemq::commands::ActiveMQMessageTemplate, 432
  - cms::Message, 2544
- getCMSTimestamp
  - activemq::commands::ActiveMQMessageTemplate, 432

- cms::Message, 2544
- getCMSType
  - activemq::commands::ActiveMQMessageTemplate, 433
  - cms::Message, 2545
- getCMSVersion
  - activemq::core::ActiveMQConnectionMetaData, 305
  - cms::ConnectionMetaData, 1386
- getCMSXPropertyNames
  - activemq::core::ActiveMQConnectionMetaData, 305
  - cms::ConnectionMetaData, 1387
- getCollisionAvoidancePercent
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
  - activemq::core::RedeliveryPolicy, 3179
- getCommand
  - activemq::commands::ControlCommand, 1494, 1495
  - activemq::wireformat::stomp::StompFrame, 3635
- getCommandId
  - activemq::commands::BaseCommand, 760
  - activemq::commands::Command, 1195
  - activemq::commands::PartialCommand, 2919
- getCommands
  - activemq::state::TransactionState, 3876
- getComponents
  - activemq::util::CompositeData, 1220
- getConnectedBrokers
  - activemq::commands::ConnectionControl, 1269
- getConnection
  - activemq::commands::Message, 2522
  - activemq::core::ActiveMQSession, 527
- getConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1155
- getConnectionId
  - activemq::commands::BrokerInfo, 899
  - activemq::commands::ConnectionError, 1297, 1298
  - activemq::commands::ConnectionInfo, 1357
  - activemq::commands::ConsumerId, 1432
  - activemq::commands::DestinationInfo, 1729
  - activemq::commands::LocalTransactionId, 2349
  - activemq::commands::ProducerId, 3069
  - activemq::commands::RemoveSubscriptionInfo, 3224
  - activemq::commands::SessionId, 3381
  - activemq::commands::TransactionInfo, 3848, 3849
- activemq::core::ActiveMQConnection, 281
- getConnectionInfo
  - activemq::core::ActiveMQConnection, 281
- getConsumerId
  - activemq::commands::ConsumerControl, 1403
  - activemq::commands::ConsumerInfo, 1462
  - activemq::commands::MessageAck, 2561
  - activemq::commands::MessageDispatch, 2595, 2596
  - activemq::commands::MessageDispatchNotification, 2631, 2632
  - activemq::commands::MessagePull, 2740, 2741
  - activemq::core::ActiveMQConsumer, 314
  - activemq::core::DispatchData, 1786
- getConsumerInfo
  - activemq::core::ActiveMQConsumer, 315
- getConsumerState
  - activemq::state::SessionState, 3438
- getConsumerStates
  - activemq::state::SessionState, 3438
- getContent
  - activemq::commands::Message, 2522, 2523
- getContext
  - decaf::internal::net::ssl::DefaultSSLContext, 1691
- getCorrelationId
  - activemq::commands::Message, 2523
  - activemq::commands::MessagePull, 2741
  - activemq::commands::Response, 3286
- getCount
  - decaf::util::concurrent::CountDownLatch, 1523
- getData
  - activemq::commands::DataArrayResponse, 1529, 1530
  - activemq::commands::DataResponse, 1584, 1585
  - activemq::commands::PartialCommand, 2920
- getDataStructure
  - activemq::commands::Message, 2523
- getDataStructureType
  - activemq::commands::ActiveMQBlobMessage, 206
  - activemq::commands::ActiveMQBytesMessage, 238
  - activemq::commands::ActiveMQDestination, 326
  - activemq::commands::ActiveMQMapMessage, 365

- activemq::commands::ActiveMQMessage, 399
- activemq::commands::ActiveMQObjectMessage, 445
- activemq::commands::ActiveMQQueue, 485
- activemq::commands::ActiveMQStreamMessage, 541
- activemq::commands::ActiveMQTempDestination, 578
- activemq::commands::ActiveMQTempQueue, 607
- activemq::commands::ActiveMQTempTopic, 636
- activemq::commands::ActiveMQTextMessage, 664
- activemq::commands::ActiveMQTopic, 693
- activemq::commands::BrokerError, 865
- activemq::commands::BrokerId, 870
- activemq::commands::BrokerInfo, 899
- activemq::commands::ConnectionControl, 1269
- activemq::commands::ConnectionError, 1298
- activemq::commands::ConnectionId, 1329
- activemq::commands::ConnectionInfo, 1357
- activemq::commands::ConsumerControl, 1403
- activemq::commands::ConsumerId, 1432
- activemq::commands::ConsumerInfo, 1462
- activemq::commands::ControlCommand, 1495
- activemq::commands::DataArrayResponse, 1530
- activemq::commands::DataResponse, 1585
- activemq::commands::DataStructure, 1662
- activemq::commands::DestinationInfo, 1729
- activemq::commands::DiscoveryEvent, 1760
- activemq::commands::ExceptionResponse, 1840
- activemq::commands::FlushCommand, 1938
- activemq::commands::IntegerResponse, 2092
- activemq::commands::JournalQueueAck, 2157
- activemq::commands::JournalTopicAck, 2185
- activemq::commands::JournalTrace, 2213
- activemq::commands::JournalTransaction, 2240
- activemq::commands::KeepAliveInfo, 2267
- activemq::commands::LastPartialCommand, 2302
- activemq::commands::LocalTransactionId, 2349
- activemq::commands::Message, 2523
- activemq::commands::MessageAck, 2561
- activemq::commands::MessageDispatch, 2596
- activemq::commands::MessageDispatchNotification, 2632
- activemq::commands::MessageId, 2665
- activemq::commands::MessagePull, 2741
- activemq::commands::NetworkBridgeFilter, 2794
- activemq::commands::PartialCommand, 2920
- activemq::commands::ProducerAck, 3037
- activemq::commands::ProducerId, 3069
- activemq::commands::ProducerInfo, 3098
- activemq::commands::RemoveInfo, 3195
- activemq::commands::RemoveSubscriptionInfo, 3224
- activemq::commands::ReplayCommand, 3252
- activemq::commands::Response, 3287
- activemq::commands::SessionId, 3381
- activemq::commands::SessionInfo, 3409
- activemq::commands::ShutdownInfo, 3471
- activemq::commands::SubscriptionInfo, 3673
- activemq::commands::TransactionId, 3821
- activemq::commands::TransactionInfo, 3849
- activemq::commands::WireFormatInfo, 3986
- activemq::commands::XATransactionId, 4033
- activemq::wireformat::openwire::marshal::DataStreamMarshal, 1617
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 214
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesI, 254
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 379
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 406
- activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 451
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 496
- activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 557

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2372	activemq::wireformat::openwire::marshal::v1::LocalTransactionMarshaller	
614			
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	2372	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	
647			
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2582	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	
676			
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2623	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	
704			
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2652	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	
881			
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2689	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	
913			
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2761	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	
1281			
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2817	activemq::wireformat::openwire::marshal::v1::NetworkBridgeMarshaller	
1313			
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2943	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	
1344			
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	3061	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	
1374			
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	3093	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	
1419			
activemq::wireformat::openwire::marshal::v1::ConsumerGroupWithMarshaller	3110	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	
1448			
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	3211	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	
1481			
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3228	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	
1510			
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	3259	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	
1544			
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3316	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	
1607			
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3404	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	
1745			
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3420	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	
1779			
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	3482	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	
1863			
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3681	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	
1957			
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3856	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	
2111			
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	4009	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	
2180			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	4049	activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaller	
2209			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	222	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	
2232			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	270	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	
2263			
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	391	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	
2290			
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	418	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	
2325			
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	463	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	



activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2274	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	2274
508		activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	2313
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller	2313	activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller	2356
569		activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2570
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	2356	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2607
626		activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2640
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	2570	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
655		activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2745
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2607	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFactory	2797
688		activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2927
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	2640	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	3041
716		activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	3073
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2669	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	3106
893		activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	3199
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2745	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	3236
925		activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	3263
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2797	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	3301
1293		activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	3384
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2927	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	3428
1301		activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	3478
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	3041	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3697
1332		activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3872
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	3073	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	4001
1362		activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	4041
activemq::wireformat::openwire::marshal::v2::ConsumerGroupWithMarshaller	3106	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	210
1407		activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	250
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	3199	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	375
1436			
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3236		
1469			
activemq::wireformat::openwire::marshal::v2::ContainerCommandMarshaller	3263		
1498			
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3301		
1532			
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3384		
1595			
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3428		
1733			
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	3478		
1767			
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3697		
1847			
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3872		
1945			
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	4001		
2099			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	4041		
2164			
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	210		
2193			
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	250		
2216			
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	375		
2247			

activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2220	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	2220
402		activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2251
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2251	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	2278
447		492	2278
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamingMessageMarshaller	2309	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	2360
492		553	2360
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2574	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	2574
610		639	2611
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2644	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2644
639		668	2681
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2753	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2753
668		696	2809
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	3049	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	3049
696		873	3081
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	3118	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	3118
873		905	3207
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	3232	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	3232
905		1273	3267
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	3311	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFormatMarshaller	3311
1273		1305	3400
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	3424	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	3424
1305		1336	3490
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	3677	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	3677
1336		1366	3860
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	4013	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	4013
1366		1411	4053
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWithMarshaller	218	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	218
1411		1440	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller		1473	
1440		1502	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller		1536	
1473		1599	
activemq::wireformat::openwire::marshal::v3::ContainerCommandMarshaller		1737	
1502		1771	
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller		1851	
1536		1949	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller		2103	
1599		2172	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller		2197	
1737			
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller			
1771			
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller			
1851			
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller			
1949			
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller			
2103			
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller			
2172			
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller			
2197			

activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	2176
258	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	2205
383	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2228
410	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2259
455	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	2282
500	
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceMessageMarshaller	2321
561	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2368
618	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2578
643	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2619
672	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2648
700	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2673
877	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2757
909	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2813
1277	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2939
1309	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	3045
1340	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	3077
1370	
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	3102
1415	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	3219
1444	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	3248
1477	
activemq::wireformat::openwire::marshal::v4::ContainerProviderMarshaller	3255
1506	
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	3296
1540	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	3388
1603	
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	3432
1741	
activemq::wireformat::openwire::marshal::v4::DiscardExpiryMarshaller	3494
1775	
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	3689
1859	
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	3868
1953	
activemq::wireformat::openwire::marshal::v4::IntegrationResponseMarshaller	4005
2107	
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFailureMarshaller	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller	
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1961
4045	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireMessageMarshaller	2115
226	
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireMessageMarshaller	2168
262	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapWireMessageMarshaller	2189
387	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2236
414	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireMessageMarshaller	2255
459	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueWireMarshaller	2286
504	
activemq::wireformat::openwire::marshal::v5::ActiveMQStackWireMessageMarshaller	2317
565	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2364
622	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2586
651	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextWireMessageMarshaller	2615
680	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2656
708	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2677
885	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2749
917	
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2805
1285	
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2931
1317	
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	3053
1348	
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	3085
1378	
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	3114
1423	
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	3215
1452	
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	3244
1485	
activemq::wireformat::openwire::marshal::v5::ContainerCommandMarshaller	3275
1514	
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller	3306
1548	
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	3396
1587	
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	3416
1753	
activemq::wireformat::openwire::marshal::v5::DiscardQueueMarshaller	3486
1783	
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	3685
1855	

activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1763
3852	
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	1843
3993	
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	1941
4057	
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	2095
230	
activemq::wireformat::openwire::marshal::v6::ActiveMQByteWireFormatMarshaller	2160
266	
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	2201
395	
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	2224
422	
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	2243
467	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormatMarshaller	2270
512	
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	2305
573	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	2352
630	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	2566
659	
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	2627
684	
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	2636
712	
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	2685
889	
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	2765
921	
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	2801
1289	
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	2923
1321	
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	3057
1352	
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	3089
1382	
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	3122
1427	
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	3203
1456	
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	3240
1489	
activemq::wireformat::openwire::marshal::v6::ContentCommandMarshaller	3271
1518	
activemq::wireformat::openwire::marshal::v6::DataActiveResponseMarshaller	3321
1552	
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	3392
1591	
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	3412
1749	

- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3279
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::CmsTemplate::ResolveReceiveExecutor, 3474
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 1403, 1404
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::ConsumerControl, 3693
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 1463
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::ConsumerInfo, 1462, 3864
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 1729, 1730
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::DestinationInfo, 3997
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 2157, 2158
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::JournalQueueAck, 4037
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::JournalTopicAck, 2185, 2186
- getDefault
  - decaf::net::ServerSocketFactory, 3363
  - decaf::net::SocketFactory, 3527
  - decaf::net::ssl::SSLContext, 3545
  - decaf::net::ssl::SSLServerSocketFactory, 3561
  - decaf::net::ssl::SSLSocketFactory, 3572
- getDefaultBacklog
  - decaf::net::ServerSocket, 3357
- getDefaultCipherSuites
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1695
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1702
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2856
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2880
  - decaf::net::ssl::SSLServerSocketFactory, 3561
  - decaf::net::ssl::SSLSocketFactory, 3572
- getDefaultDestination
  - activemq::cmsutil::CmsTemplate, 1175
- getDefaultDestinationName
  - activemq::cmsutil::CmsTemplate, 1175
- getDefaultSSLParameters
  - decaf::net::ssl::SSLContext, 3546
- getDelay
  - decaf::util::concurrent::Delayed, 1720
- getDeliveryMode
  - activemq::cmsutil::CachedProducer, 1076
  - activemq::cmsutil::CmsTemplate, 1175
  - activemq::core::ActiveMQProducer, 472
  - cms::MessageProducer, 2726
- getDeliverySequenceId
  - activemq::commands::MessageDispatchNotification, 2632
- getDestination
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3066
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
  - activemq::cmsutil::CmsTemplate::ResolveProducer, 3278
- activemq::commands::JournalTopicAck, 2185, 2186
- activemq::commands::Message, 2523, 2524
- activemq::commands::MessageAck, 2561, 2562
- activemq::commands::MessageDispatch, 2596
- activemq::commands::MessageDispatchNotification, 2632
- activemq::commands::MessagePull, 2741, 2742
- activemq::commands::ProducerInfo, 3098
- activemq::commands::SubscriptionInfo, 3673, 3674
- activemq::core::ActiveMQDestination, 327
- activemq::core::ActiveMQQueue, 485
- activemq::core::ActiveMQTempQueue, 607
- activemq::core::ActiveMQTempTopic, 636
- activemq::core::ActiveMQTopic, 693
- cms::Destination, 1724
- getDisableMessageID
  - activemq::cmsutil::CachedProducer, 1077
  - activemq::core::ActiveMQProducer, 472
  - cms::MessageProducer, 2727
- getDisableMessageTimeStamp
  - activemq::cmsutil::CachedProducer, 1077
  - activemq::core::ActiveMQProducer, 472
- cms::MessageProducer, 2727
- getDouble
  - activemq::commands::ActiveMQMapMessage, 2475
  - activemq::util::PrimitiveList, 2984
  - activemq::util::PrimitiveMap, 2995
  - activemq::util::PrimitiveValueNode, 3019
  - decaf::internal::nio::ByteArrayBuffer, 1009

- decaf::internal::util::ByteArrayAdapter, 978
- decaf::nio::ByteBuffer, 1042, 1043
- getDoubleArray
  - decaf::internal::util::ByteArrayAdapter, 978
- getDoubleAt
  - decaf::internal::util::ByteArrayAdapter, 978
- getDoubleCapacity
  - decaf::internal::util::ByteArrayAdapter, 978
- getDoubleProperty
  - activemq::commands::ActiveMQMessageTemplate, 433
  - activemq::wireformat::openwire::utils::MessageProperty, 2734
  - cms::Message, 2545
- getDurableTopicPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
  - activemq::core::PrefetchPolicy, 2977
- getEnabledCipherSuites
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2845
  - decaf::internal::net::ssl::openssl::OpenSSLServiceSocket, 2849
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2863
  - decaf::net::ssl::SSLServerSocket, 3556
  - decaf::net::ssl::SSLSocket, 3566
- getEnabledProtocols
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2849
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2864
  - decaf::net::ssl::SSLServerSocket, 3557
  - decaf::net::ssl::SSLSocket, 3566
- getEncoded
  - decaf::security::auth::x500::X500Principal, 4027
  - decaf::security::cert::Certificate, 1087
  - decaf::security::Key, 2294
- getEnumeration
  - activemq::core::ActiveMQQueueBrowser, 488
  - cms::QueueBrowser, 3153
- getenv
  - decaf::lang::System, 3730
- getErrorCode
  - decaf::net::SocketError, 3520
- getErrorManager
  - decaf::util::logging::Handler, 1979
- getErrorString
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2874
  - decaf::net::SocketError, 3520
- getException
  - activemq::commands::ConnectionError, 1298
  - activemq::commands::ExceptionResponse, 1841
- getExceptionClass
  - activemq::commands::BrokerError, 865
- getExceptionListener
  - activemq::core::ActiveMQConnection, 281
  - activemq::core::ActiveMQConnectionFactory, 287
  - activemq::core::ActiveMQSession, 527
  - cms::Connection, 1264
- getExpiration
  - activemq::commands::Message, 2524
- getFileDescriptor
  - decaf::net::SocketImpl, 3532
- getFilter
  - decaf::util::logging::Handler, 1979
  - decaf::util::logging::Logger, 2392
- getFirstMessageId
  - activemq::commands::MessageAck, 2562
- getFirstNakNumber
  - activemq::commands::ReplayCommand, 3253
- getFloat
  - activemq::commands::ActiveMQMapMessage, 366
  - activemq::util::PrimitiveList, 2985
  - activemq::util::PrimitiveMap, 2995
  - activemq::util::PrimitiveValueNode, 3020
  - cms::MapMessage, 2476
- getFloatArray
  - decaf::internal::nio::ByteBuffer, 1009, 1010
  - decaf::internal::util::ByteArrayAdapter, 979
  - decaf::nio::ByteBuffer, 1043
- getFloatAt
  - decaf::internal::util::ByteArrayAdapter, 979
- getFloatCapacity
  - decaf::internal::util::ByteArrayAdapter, 980
- getFloatProperty
  - activemq::commands::ActiveMQMessageTemplate, 433

- activemq::wireformat::openwire::utils::MessageProducerState, 1492
- 2734
- cms::Message, 2546
- getFormat
  - decaf::security::Key, 2294
- getFormatId
  - activemq::commands::XATransactionId, 4033
- getFormatter
  - decaf::util::logging::Handler, 1979
- getFragment
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3954
  - decaf::net::URI, 3925
- getFreeThreadCount
  - decaf::util::concurrent::ThreadPool, 3779
- getGlobalPool
  - decaf::internal::AprPool, 728
  - decaf::internal::DecafRuntime, 1670
- getGlobalTransactionId
  - activemq::commands::XATransactionId, 4034
- getGroupID
  - activemq::commands::Message, 2524
- getGroupSequence
  - activemq::commands::Message, 2524
- getHandlers
  - decaf::util::logging::Logger, 2392
- getHead
  - decaf::util::logging::Formatter, 1965
  - decaf::util::logging::XMLFormatter, 4061
- getHoldCount
  - decaf::util::concurrent::locks::ReentrantLock, 3183
- getHost
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3954
  - decaf::net::URI, 3925
- getHostAddress
  - decaf::net::InetAddress, 2019
- getHostName
  - decaf::net::InetAddress, 2019
- getHostname
  - activemq::util::IdGenerator, 1990
- getId
  - activemq::state::TransactionState, 3876
  - decaf::lang::Thread, 3769
- getIndex
  - decaf::net::URISyntaxException, 3950
- getInetAddress
  - decaf::net::Socket, 3510
  - decaf::net::SocketImpl, 3533
- getInfo
  - activemq::state::ConnectionState, 1390
- getInitialDelayTime
  - activemq::transport::inactivity::InactivityMonitor, 2005
- getInitialReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1877
- getInitialRedeliveryDelay
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
  - activemq::core::RedeliveryPolicy, 3179
- getInput
  - decaf::net::URISyntaxException, 3951
- getInputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2864
  - decaf::internal::net::tcp::TcpSocket, 3742
  - decaf::net::Socket, 3510
  - decaf::net::SocketImpl, 3533
- getInstance
  - activemq::transport::mock::MockTransport, 2771
  - activemq::transport::TransportRegistry, 3903
  - activemq::wireformat::WireFormatRegistry, 4018
  - decaf::util::concurrent::ThreadPool, 3780
  - decaf::util::logging::LogWriter, 2418
- getInt
  - activemq::commands::ActiveMQMapMessage, 366
  - activemq::util::PrimitiveList, 2985
  - activemq::util::PrimitiveMap, 2996
  - activemq::util::PrimitiveValueNode, 3020
  - cms::MapMessage, 2476
  - decaf::internal::nio::ByteBuffer, 1010
  - decaf::internal::util::ByteArrayAdapter, 980
  - decaf::nio::ByteBuffer, 1044
- getIntArray
  - decaf::internal::util::ByteArrayAdapter, 980
- getIntAt
  - decaf::internal::util::ByteArrayAdapter, 980
- getIntCapacity
  - decaf::internal::util::ByteArrayAdapter, 981
- getIntProperty
  - activemq::commands::ActiveMQMessageTemplate, 434



- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2735
- cms::Message, 2546
- getIssuerUniqueID
  - decaf::security::cert::X509Certificate, 4029
- getIssuerX500Principal
  - decaf::security::cert::X509Certificate, 4029
- getKeepAlive
  - decaf::net::Socket, 3510
- getKey
  - decaf::util::Map::Entry, 1824
- getKeyUsage
  - decaf::security::cert::X509Certificate, 4029
- getLastDeliveredSequenceId
  - activemq::commands::RemoveInfo, 3195
  - activemq::core::ActiveMQConsumer, 315
  - activemq::core::ActiveMQSession, 527
- getLastMessageId
  - activemq::commands::MessageAck, 2562
- getLastNakNumber
  - activemq::commands::ReplayCommand, 3253
- getLastSequenceId
  - activemq::util::LongSequenceGenerator, 2455
- getLeastSignificantBits
  - decaf::util::UUID, 3972
- getLevel
  - decaf::util::logging::Handler, 1979
  - decaf::util::logging::Logger, 2392
  - decaf::util::logging::LogRecord, 2414
- getLimit
  - activemq::util::MemoryUsage, 2513
- getList
  - activemq::util::PrimitiveValueNode, 3020
- getLocalAddress
  - decaf::internal::net::tcp::TcpSocket, 3743
  - decaf::net::Socket, 3510
  - decaf::net::SocketImpl, 3533
- getLocalHost
  - decaf::net::InetAddress, 2020
- getLocalPort
  - decaf::net::ServerSocket, 3357
  - decaf::net::Socket, 3511
  - decaf::net::SocketImpl, 3533
- getLogger
  - decaf::util::logging::Logger, 2392
  - decaf::util::logging::LogManager, 2409
- getLoggerName
  - decaf::util::logging::LogRecord, 2414
- getLoggerNames
  - decaf::util::logging::LogManager, 2409
- getLogManager
  - decaf::util::logging::LogManager, 2409
- getLong
  - activemq::commands::ActiveMQMapMessage, 367
  - activemq::util::PrimitiveList, 2985
  - activemq::util::PrimitiveMap, 2996
  - activemq::util::PrimitiveValueNode, 3020
  - cms::MapMessage, 2476
  - decaf::internal::nio::ByteBuffer, 1011
  - decaf::internal::util::ByteArrayAdapter, 981
  - decaf::nio::ByteBuffer, 1044, 1045
- getLongArray
  - decaf::internal::util::ByteArrayAdapter, 981
- getLongAt
  - decaf::internal::util::ByteArrayAdapter, 981
- getLongCapacity
  - decaf::internal::util::ByteArrayAdapter, 982
- getLongProperty
  - activemq::commands::ActiveMQMessageTemplate, 434
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2735
  - cms::Message, 2547
- getLoopbackAddress
  - decaf::net::InetAddress, 2020
- getMagic
  - activemq::commands::WireFormatInfo, 3986
- getManaged
  - decaf::internal::util::GenericResource, 1974
- getMap
  - activemq::commands::ActiveMQMapMessage, 367
  - activemq::util::PrimitiveValueNode, 3021
- getMapNames
  - activemq::commands::ActiveMQMapMessage, 367
  - cms::MapMessage, 2477
- getMarshaledForm
  - activemq::commands::BaseDataStructure, 832
  - activemq::wireformat::MarshalAware, 2485
- getMarshaledProperties
  - activemq::commands::Message, 2524
  - activemq::commands::WireFormatInfo, 3986
- getMaxCacheSize
  - activemq::state::ConnectionStateTracker, 1394
  - activemq::transport::failover::FailoverTransport, 1877

- getMaximumPendingMessageLimit
  - activemq::commands::ConsumerInfo, 1463
- getMaximumRedeliveries
  - activemq::core::policies::DefaultRedeliveryPolicy, 1678
  - activemq::core::RedeliveryPolicy, 3179
- getMaxInactivityDuration
  - activemq::commands::WireFormatInfo, 3986
  - activemq::wireformat::openwire::OpenWireFormat, 2891
- getMaxInactivityDurationInitialDelay
  - activemq::commands::WireFormatInfo, 3986
- getMaxInactivityDurationInitialDelay
  - activemq::wireformat::openwire::OpenWireFormat, 2891
- getMaxPrefetchLimit
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
  - activemq::core::PrefetchPolicy, 2978
- getMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1877
- getMaxReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1877
- getMaxThreads
  - decaf::util::concurrent::ThreadPool, 3780
- getMessage
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
  - activemq::commands::BrokerError, 865
  - activemq::commands::JournalTrace, 2213, 2214
  - activemq::commands::MessageDispatch, 2596
  - activemq::core::DispatchData, 1786
  - cms::CMSException, 1161
  - decaf::lang::Exception, 1835
  - decaf::lang::Throwable, 3785
  - decaf::util::logging::LogRecord, 2414
- getMessageAck
  - activemq::commands::JournalQueueAck, 2158
- getMessageAvailableCount
  - activemq::core::ActiveMQConsumer, 315
- getMessageCount
  - activemq::commands::MessageAck, 2562
- getMessageId
  - activemq::commands::JournalTopicAck, 2186
  - activemq::commands::Message, 2524
- activemq::commands::MessageDispatchNotification, 2632
- activemq::commands::MessagePull, 2742
- getMessageListener
  - activemq::cmsutil::CachedConsumer, 1072
  - activemq::core::ActiveMQConsumer, 315
  - cms::MessageConsumer, 2590
- getMessageProperties
  - activemq::commands::Message, 2524
- getMessageSelector
  - activemq::cmsutil::CachedConsumer, 1072
  - activemq::core::ActiveMQConsumer, 315
  - activemq::core::ActiveMQQueueBrowser, 488
  - cms::MessageConsumer, 2590
  - cms::QueueBrowser, 3154
- getMessageSequenceId
  - activemq::commands::JournalTopicAck, 2186
- getMetaData
  - activemq::core::ActiveMQConnection, 281
  - cms::Connection, 1264
- getMimeType
  - activemq::commands::ActiveMQBlobMessage, 206
- getMostSignificantBits
  - decaf::util::UUID, 3972
- getName
  - activemq::commands::ActiveMQBlobMessage, 206
  - decaf::lang::Thread, 3769
  - decaf::security::auth::x500::X500Principal, 4027
  - decaf::security::Principal, 3026
  - decaf::util::logging::Level, 2334
  - decaf::util::logging::Logger, 2392
- getNeedClientAuth
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2850
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2864
  - decaf::net::ssl::SSLParameters, 3552
  - decaf::net::ssl::SSLServerSocket, 3557
  - decaf::net::ssl::SSLSocket, 3566
- getNetworkBrokerId
  - activemq::commands::NetworkBridgeFilter, 2794, 2795
- getNetworkConsumerPath
  - activemq::commands::ConsumerInfo, 1463
- getNetworkProperties
  - activemq::commands::BrokerInfo, 899, 900
- getNetworkRuntime

- decaf::internal::net::Network, 2791
- getNetworkTTL
  - activemq::commands::NetworkBridgeFilter, 2795
- getNextConsumerId
  - activemq::core::ActiveMQSession, 527
- getNextLocalTransactionId
  - activemq::core::ActiveMQConnection, 282
- getNextProducerId
  - activemq::core::ActiveMQSession, 527
- getNextSequenceId
  - activemq::util::LongSequenceGenerator, 2455
- getNextSessionId
  - activemq::core::ActiveMQConnection, 282
- getNextTempDestinationId
  - activemq::core::ActiveMQConnection, 282
- getNotAfter
  - decaf::security::cert::X509Certificate, 4029
- getNotBefore
  - decaf::security::cert::X509Certificate, 4029
- getNumReceivedMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2771
- getNumReceivedMessages
  - activemq::transport::mock::MockTransport, 2771
- getNumSentKeepAlives
  - activemq::transport::mock::MockTransport, 2771
- getNumSentKeepAlivesBeforeFail
  - activemq::transport::mock::MockTransport, 2771
- getNumSentMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2771
- getNumSentMessages
  - activemq::transport::mock::MockTransport, 2771
- getObjectId
  - activemq::commands::RemoveInfo, 3196
- getOOBInline
  - decaf::net::Socket, 3511
- getOperationType
  - activemq::commands::DestinationInfo, 1730
- getOption
  - decaf::internal::net::tcp::TcpSocket, 3743
  - decaf::net::SocketImpl, 3533
- getOptions
  - activemq::commands::ActiveMQDestination, 327
- getOrderedTarget
  - activemq::commands::ActiveMQDestination, 327
- getOriginalDestination
  - activemq::commands::Message, 2524, 2525
- getOriginalTransactionId
  - activemq::commands::Message, 2525
- getOutputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2864
  - decaf::internal::net::tcp::TcpSocket, 3743
  - decaf::net::Socket, 3511
  - decaf::net::SocketImpl, 3534
- getParameters
  - activemq::util::CompositeData, 1220
- getParent
  - decaf::util::logging::Logger, 2393
- getParentId
  - activemq::commands::ConsumerId, 1432
  - activemq::commands::ProducerId, 3069
  - activemq::commands::SessionId, 3381
- getPassword
  - activemq::commands::ConnectionInfo, 1358
  - activemq::core::ActiveMQConnection, 282
  - activemq::core::ActiveMQConnectionFactory, 297
- getPath
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3954
  - decaf::net::URI, 3926
- getPeerBrokerInfos
  - activemq::commands::BrokerInfo, 900
- getPhysicalName
  - activemq::commands::ActiveMQDestination, 327
- getPooledThreadListener
  - decaf::util::concurrent::PooledThread, 2969
- getPoolSize
  - decaf::util::concurrent::ThreadPool, 3780
- getPort
  - decaf::internal::net::URIType, 3954
  - decaf::net::Socket, 3511
  - decaf::net::SocketImpl, 3534
  - decaf::net::URI, 3926
- getPreferredWireFormatInfo
  - activemq::wireformat::openwire::OpenWireFormat, 2891
- getPrefetch
  - activemq::commands::ConsumerControl, 1404
- getPrefetchPolicy
  - activemq::core::ActiveMQConnection, 283
  - activemq::core::ActiveMQConnectionFactory, 297

- getPrefetchSize
  - activemq::commands::ConsumerInfo, 1463
- getPreparedResult
  - activemq::state::TransactionState, 3876
- getPriority
  - activemq::cmsutil::CachedProducer, 1077
  - activemq::cmsutil::CmsTemplate, 1175
  - activemq::commands::ConsumerInfo, 1463
  - activemq::commands::Message, 2525
  - activemq::core::ActiveMQProducer, 473
  - cms::MessageProducer, 2727
  - decaf::lang::Thread, 3769
- getProducerId
  - activemq::commands::Message, 2525
  - activemq::commands::MessageId, 2665, 2666
  - activemq::commands::ProducerAck, 3037, 3038
  - activemq::commands::ProducerInfo, 3098
  - activemq::core::ActiveMQProducer, 473
- getProducerInfo
  - activemq::core::ActiveMQProducer, 473
- getProducerSequenceId
  - activemq::commands::MessageId, 2666
- getProducerState
  - activemq::state::SessionState, 3438
- getProducerStates
  - activemq::state::SessionState, 3438
  - activemq::state::TransactionState, 3876
- getProducerWindowSize
  - activemq::core::ActiveMQConnection, 283
  - activemq::core::ActiveMQConnectionFactory, 297
- getProperties
  - activemq::commands::WireFormatInfo, 3987
  - activemq::util::ActiveMQProperties, 479, 480
  - activemq::wireformat::stomp::StompFrame, 3635
  - decaf::lang::System, 3731
  - decaf::util::logging::LogManager, 2410
- getProperty
  - activemq::util::ActiveMQProperties, 480
  - activemq::wireformat::stomp::StompFrame, 3636
  - cms::CMSProperties, 1166
  - decaf::lang::System, 3731
  - decaf::util::logging::LogManager, 2410
  - decaf::util::Properties, 3128, 3129
- getPropertyNames
  - activemq::commands::ActiveMQMessageTemplate, 434
  - cms::Message, 2547
- getProtocols
  - decaf::net::ssl::SSLParameters, 3552
- getProviderMajorVersion
  - activemq::core::ActiveMQConnectionMetaData, 305
  - cms::ConnectionMetaData, 1387
- getProviderMinorVersion
  - activemq::core::ActiveMQConnectionMetaData, 305
  - cms::ConnectionMetaData, 1387
- getProviderVersion
  - activemq::core::ActiveMQConnectionMetaData, 306
  - cms::ConnectionMetaData, 1388
- getPublicKey
  - decaf::security::cert::Certificate, 1087
- getQuery
  - decaf::internal::net::URIType, 3955
  - decaf::net::URI, 3926
- getQueue
  - activemq::core::ActiveMQQueueBrowser, 489
  - cms::QueueBrowser, 3154
- getQueueBrowserPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1674
  - activemq::core::PrefetchPolicy, 2978
- getQueueName
  - activemq::commands::ActiveMQQueue, 486
  - activemq::commands::ActiveMQTempQueue, 607
  - cms::Queue, 3148
  - cms::TemporaryQueue, 3759
- getQueuePrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1675
  - activemq::core::PrefetchPolicy, 2978
- getRawAuthority
  - decaf::net::URI, 3926
- getRawFragment
  - decaf::net::URI, 3926
- getRawPath
  - decaf::net::URI, 3926
- getRawQuery
  - decaf::net::URI, 3927
- getRawSchemeSpecificPart
  - decaf::net::URI, 3927
- getRawUserInfo
  - decaf::net::URI, 3927
- getReadCheckTime
  - activemq::transport::inactivity::InactivityMonitor, 2005
- getReason

- decaf::net::URISyntaxException, 3951
- getReceiveBufferSize
  - decaf::net::ServerSocket, 3357
  - decaf::net::Socket, 3511
- getReceiveTimeout
  - activemq::cmsutil::CmsTemplate, 1175
- getReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1877
- getReconnectTo
  - activemq::commands::ConnectionControl, 1269, 1270
- getRecoveringPullConsumers
  - activemq::state::ConnectionState, 1390
- getRedeliveryCounter
  - activemq::commands::Message, 2525
  - activemq::commands::MessageDispatch, 2596
- getRedeliveryDelay
  - activemq::core::policies::DefaultRedeliveryPolicy, 1679
  - activemq::core::RedeliveryPolicy, 3179
- getRedeliveryPolicy
  - activemq::core::ActiveMQConnection, 283
  - activemq::core::ActiveMQConnectionFactory, 298
  - activemq::core::ActiveMQConsumer, 316
- getRemaining
  - decaf::util::zip::Inflater, 2030
- getRemoteAddress
  - activemq::transport::failover::FailoverTransport, 1877
  - activemq::transport::IOTransport, 2147
  - activemq::transport::mock::MockTransport, 2771
  - activemq::transport::Transport, 3884
  - activemq::transport::TransportFilter, 3894
- getRemoteBlobUrl
  - activemq::commands::ActiveMQBlobMessage, 206
- getReplyTo
  - activemq::commands::Message, 2525
- getResourceLifecycleManager
  - activemq::cmsutil::CmsAccessor, 1155
  - activemq::cmsutil::SessionPool, 3436
- getResponse
  - activemq::transport::correlator::FutureResponse, 1969, 1970
- getResult
  - activemq::commands::IntegerResponse, 2093
- getReuseAddress
  - decaf::net::ServerSocket, 3357
  - decaf::net::Socket, 3512
- getRuntime
  - decaf::lang::Runtime, 3326
- getRuntimeLock
  - decaf::internal::net::Network, 2791
- getSafeValue
  - decaf::util::StlQueue, 3618
- getScheme
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3955
  - decaf::net::URI, 3927
- getSchemeSpecificPart
  - decaf::internal::net::URIType, 3955
  - decaf::net::URI, 3927
- getSeedFromId
  - activemq::util::IdGenerator, 1990
- getSelector
  - activemq::commands::ConsumerInfo, 1463
  - activemq::commands::SubscriptionInfo, 3674
- getSendBufferSize
  - decaf::net::Socket, 3512
- getSendTimeout
  - activemq::core::ActiveMQConnection, 283
  - activemq::core::ActiveMQConnectionFactory, 298
  - activemq::core::ActiveMQProducer, 473
- getSequenceFromId
  - activemq::util::IdGenerator, 1990
- getServerSocketFactory
  - decaf::net::ssl::SSLContext, 3546
- getServiceName
  - activemq::commands::DiscoveryEvent, 1760
- getSession
  - activemq::cmsutil::PooledSession, 2965
- getSessionAcknowledgeMode
  - activemq::cmsutil::CmsAccessor, 1155
- getSessionId
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionInfo, 3409
  - activemq::core::ActiveMQSession, 527
- getSessionInfo
  - activemq::core::ActiveMQSession, 528
- getSessionState
  - activemq::state::ConnectionState, 1390
- getSessionStates
  - activemq::state::ConnectionState, 1390
- getShort
  - activemq::commands::ActiveMQMapMessage, 367
  - activemq::util::PrimitiveList, 2986
  - activemq::util::PrimitiveMap, 2997
  - activemq::util::PrimitiveValueNode, 3021

- cms::MapMessage, 2477
- decaf::internal::nio::ByteBuffer, 1011, 1012
- decaf::internal::util::ByteArrayAdapter, 982
- decaf::nio::ByteBuffer, 1045
- getShortArray
  - decaf::internal::util::ByteArrayAdapter, 982
- getShortAt
  - decaf::internal::util::ByteArrayAdapter, 983
- getShortCapacity
  - decaf::internal::util::ByteArrayAdapter, 983
- getShortProperty
  - activemq::commands::ActiveMQMessageTemplate, 435
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2735
  - cms::Message, 2548
- getSigAlgName
  - decaf::security::cert::X509Certificate, 4029
- getSigAlgOID
  - decaf::security::cert::X509Certificate, 4029
- getSigAlgParams
  - decaf::security::cert::X509Certificate, 4029
- getSignature
  - decaf::security::cert::X509Certificate, 4029
- getSize
  - activemq::commands::ActiveMQTextMessage, 664
  - activemq::commands::Message, 2525
  - activemq::commands::ProducerAck, 3038
- getSocketFactory
  - decaf::net::ssl::SSLContext, 3546
- getSocketHandle
  - decaf::internal::net::tcp::TcpSocket, 3744
- getSoLinger
  - decaf::net::Socket, 3512
- getSoTimeout
  - decaf::net::ServerSocket, 3358
  - decaf::net::Socket, 3512
- getSource
  - decaf::internal::net::URIType, 3955
- getSourceFile
  - decaf::util::logging::LogRecord, 2415
- getSourceFunction
  - decaf::util::logging::LogRecord, 2415
- getSourceLine
  - decaf::util::logging::LogRecord, 2415
- getSSLParameters
  - decaf::net::ssl::SSLSocket, 3567
- getStackTrace
  - cms::CMSEException, 1161
  - decaf::lang::Exception, 1835
  - decaf::lang::Throwable, 3785
- getStackTraceElements
  - activemq::commands::BrokerError, 865
- getStackTraceString
  - cms::CMSEException, 1162
  - decaf::lang::Exception, 1835
  - decaf::lang::Throwable, 3785
- getStartupMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1877
- getState
  - decaf::lang::Thread, 3769
- getString
  - activemq::commands::ActiveMQMapMessage, 368
  - activemq::util::PrimitiveList, 2986
  - activemq::util::PrimitiveMap, 2997
  - activemq::util::PrimitiveValueNode, 3021
  - cms::MapMessage, 2477
- getStringProperty
  - activemq::commands::ActiveMQMessageTemplate, 435
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2735
  - cms::Message, 2548
- getSubscriptionName
  - activemq::commands::RemoveSubscriptionInfo, 3224
  - activemq::commands::SubscriptionInfo, 3674
- getSubjectUniqueID
  - decaf::security::cert::X509Certificate, 4029
- getSubjectX500Principal
  - decaf::security::cert::X509Certificate, 4029
- getSubscribedDestination
  - activemq::commands::SubscriptionInfo, 3674
- getSubscriptionName
  - activemq::commands::ConsumerInfo, 1463
- getSubscriptionName
  - activemq::commands::JournalTopicAck, 2186
- getSupportedCipherSuites
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1696
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1702
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2850

- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2727
- 2857
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2865
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2880
- decaf::net::ssl::SSLServerSocket, 3557
- decaf::net::ssl::SSLServerSocketFactory, 3561
- decaf::net::ssl::SSLSocket, 3567
- decaf::net::ssl::SSLSocketFactory, 3573
- getSupportedProtocols
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2850
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2865
- decaf::net::ssl::SSLServerSocket, 3557
- decaf::net::ssl::SSLSocket, 3567
- getSupportedSSLParameters
- decaf::net::ssl::SSLContext, 3546
- getTail
- decaf::util::logging::Formatter, 1965
- decaf::util::logging::XMLFormatter, 4061
- getTargetConsumerId
- activemq::commands::Message, 2525, 2526
- getTBSertificate
- decaf::security::cert::X509Certificate, 4029
- getTcpNoDelay
- decaf::net::Socket, 3513
- getTempDestinations
- activemq::state::ConnectionState, 1390
- getText
- activemq::commands::ActiveMQTextMessage, 665
- cms::TextMessage, 3763
- getThrown
- decaf::util::logging::LogRecord, 2415
- getTime
- decaf::util::Date, 1667
- getTimeout
- activemq::commands::DestinationInfo, 1730
- activemq::commands::MessagePull, 2742
- activemq::transport::failover::FailoverTransport, 1877
- getTimeStamp
- activemq::commands::Message, 2526
- decaf::util::logging::LogRecord, 2415
- getTimeToLive
- activemq::cmsutil::CachedProducer, 1077
- activemq::cmsutil::CmsTemplate, 1176
- activemq::core::ActiveMQProducer, 473
- activemq::core::ActiveMQMessageProducer, 2727
- getTopicName
- activemq::commands::ActiveMQTempTopic, 636
- activemq::commands::ActiveMQTopic, 694
- cms::TemporaryTopic, 3761
- cms::Topic, 3817
- getTopicPrefetch
- activemq::core::policies::DefaultPrefetchPolicy, 1675
- activemq::core::PrefetchPolicy, 2978
- getTrafficClass
- decaf::net::Socket, 3513
- getTransactionContext
- activemq::core::ActiveMQSession, 528
- getTransactionId
- activemq::commands::JournalTopicAck, 2186
- activemq::commands::JournalTransaction, 2240, 2241
- activemq::commands::Message, 2526
- activemq::commands::MessageAck, 2562
- activemq::commands::TransactionInfo, 3849
- activemq::core::ActiveMQTransactionContext, 720
- getTransactionState
- activemq::state::ConnectionState, 1390
- activemq::state::ProducerState, 3125
- getTransactionStates
- activemq::state::ConnectionState, 1390
- getTransport
- activemq::core::ActiveMQConnection, 283
- activemq::transport::failover::BackupTransport, 752
- getTransportListener
- activemq::transport::failover::FailoverTransport, 1877
- activemq::transport::IOTransport, 2147
- activemq::transport::mock::MockTransport, 2772
- activemq::transport::Transport, 3884
- activemq::transport::TransportFilter, 3894
- getTransportNames
- activemq::transport::TransportRegistry, 3903
- getTreadId
- decaf::util::logging::LogRecord, 2415
- getType
- activemq::commands::JournalTransaction, 2241
- activemq::commands::Message, 2526
- activemq::commands::TransactionInfo, 3849

- activemq::util::PrimitiveValueNode, 3021
- decaf::security::cert::Certificate, 1087
- getUncaughtExceptionHandler
  - decaf::lang::Thread, 3770
- getUnconsumedMessages
  - activemq::core::ActiveMQSessionExecutor, 535
- getURI
  - activemq::transport::failover::URIPool, 3943
- getUri
  - activemq::transport::failover::BackupTransport, 753
- getUsage
  - activemq::util::MemoryUsage, 2513
- getClientMode
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2865
  - decaf::net::ssl::SSLSocket, 3567
- getParentHandlers
  - decaf::util::logging::Logger, 2393
- getUserID
  - activemq::commands::Message, 2526
- getUserInfo
  - decaf::internal::net::URIType, 3955
  - decaf::net::URI, 3927
- getUserName
  - activemq::commands::ConnectionInfo, 1358
- getUsername
  - activemq::core::ActiveMQConnection, 284
  - activemq::core::ActiveMQConnectionFactory, 298
- getValue
  - activemq::commands::BrokerId, 871
  - activemq::commands::ConnectionId, 1329
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::LocalTransactionId, 2349
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionId, 3382
  - activemq::util::PrimitiveValueNode, 3022
  - decaf::internal::net::SocketFileDescriptor, 3528
  - decaf::util::Map::Entry, 1824
  - decaf::util::zip::Adler32, 723
  - decaf::util::zip::Checksum, 1142
  - decaf::util::zip::CRC32, 1525
- getVersion
  - activemq::commands::WireFormatInfo, 3987
- activemq::wireformat::openwire::OpenWireFormat, 2892
- activemq::wireformat::stomp::StompWireFormat, 3644
- activemq::wireformat::WireFormat, 3977
- decaf::security::cert::X509Certificate, 4029
- getWantClientAuth
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2850
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2866
  - decaf::net::ssl::SSLParameters, 3552
  - decaf::net::ssl::SSLServerSocket, 3557
  - decaf::net::ssl::SSLSocket, 3568
- getWasPrepared
  - activemq::commands::JournalTransaction, 2241
- getWhen
  - decaf::util::TimerTask, 3802
- getWindowSize
  - activemq::commands::ProducerInfo, 3098
- getWireFormat
  - activemq::transport::mock::MockTransport, 2772
- getWireFormatNames
  - activemq::wireformat::WireFormatRegistry, 4018
- getWriteCheckTime
  - activemq::transport::inactivity::InactivityMonitor, 2005
- globalTransactionId
  - activemq::commands::XATransactionId, 4035
- good\_match
  - internal\_state, 2120
- groupID
  - activemq::commands::Message, 2532
- groupSequence
  - activemq::commands::Message, 2532
- GT\_OFF
  - gzguts.h, 4729
- GUNZIP
  - inflate.h, 4732
- GZ\_APPEND
  - gzguts.h, 4729
- gz\_header
  - zlib.h, 4742
- gz\_header\_s, 1975
  - comm\_max, 1975
  - comment, 1975
  - done, 1975
  - extra, 1975



- extra\_len, 1975
- extra\_max, 1975
- hcrc, 1975
- name, 1975
- name\_max, 1975
- os, 1975
- text, 1975
- time, 1975
- xflags, 1975
- gz\_headerp
  - zlib.h, 4742
- GZ\_NONE
  - gzguts.h, 4729
- GZ\_READ
  - gzguts.h, 4729
- gz\_state, 1976
  - direct, 1977
  - eof, 1977
  - err, 1977
  - fd, 1977
  - have, 1977
  - how, 1977
  - in, 1977
  - level, 1977
  - mode, 1977
  - msg, 1977
  - next, 1977
  - out, 1977
  - path, 1977
  - pos, 1977
  - raw, 1977
  - seek, 1977
  - size, 1977
  - skip, 1977
  - start, 1977
  - strategy, 1977
  - strm, 1977
  - want, 1977
- gz\_statep
  - gzguts.h, 4729
- GZ\_WRITE
  - gzguts.h, 4729
- GZBUFSIZE
  - gzguts.h, 4729
- gzFile
  - zlib.h, 4742
- gzguts.h
  - COPY, 4729
  - GT\_OFF, 4729
  - GZ\_APPEND, 4729
  - GZ\_NONE, 4729
  - GZ\_READ, 4729
  - gz\_statep, 4729
  - GZ\_WRITE, 4729
  - GZBUFSIZE, 4729
  - GZIP, 4729
  - local, 4729
  - LOOK, 4729
  - OF, 4729
  - ZLIB\_INTERNAL, 4729
  - zstrerror, 4729
- gzhead
  - internal\_state, 2120
- gzindex
  - internal\_state, 2120
- GZIP
  - deflate.h, 4727
  - gzguts.h, 4729
- Handler
  - decaf::util::logging::Handler, 1979
- handleTransportFailure
  - activemq::transport::failover::FailoverTransport, 1877
- hasArray
  - decaf::internal::nio::ByteBuffer, 1012
  - decaf::internal::nio::CharArrayBuffer, 1115
  - decaf::internal::nio::DoubleArrayBuffer, 1806
  - decaf::internal::nio::FloatArrayBuffer, 1922
  - decaf::internal::nio::IntArrayBuffer, 2063
  - decaf::internal::nio::LongArrayBuffer, 2441
  - decaf::internal::nio::ShortArrayBuffer, 3456
  - decaf::nio::ByteBuffer, 1046
  - decaf::nio::CharBuffer, 1128
  - decaf::nio::DoubleBuffer, 1815
  - decaf::nio::FloatBuffer, 1931
  - decaf::nio::IntBuffer, 2072
  - decaf::nio::LongBuffer, 2450
  - decaf::nio::ShortBuffer, 3465
- hash\_bits
  - internal\_state, 2120
- hash\_mask
  - internal\_state, 2120
- hash\_shift
  - internal\_state, 2120
- hash\_size
  - internal\_state, 2120
- hashCode
  - decaf::security::auth::x500::X500Principal, 4027
- hasMoreMessages
  - activemq::core::ActiveMQQueueBrowser, 489
  - cms::MessageEnumeration, 2659
- hasMoreTokens
  - decaf::util::StringTokenizer, 3669

- hasNegotiator
  - activemq::wireformat::openwire::OpenWireFormat, 2892
  - activemq::wireformat::stomp::StompWireFormat, 3644
  - activemq::wireformat::WireFormat, 3977
- hasNext
  - decaf::util::Iterator, 2154
- hasPrevious
  - decaf::util::ListIterator, 2345
- hasProperty
  - activemq::util::ActiveMQProperties, 480
  - activemq::wireformat::stomp::StompFrame, 3636
  - cms::CMSProperties, 1167
  - decaf::util::Properties, 3129
- hasRemaining
  - decaf::nio::Buffer, 931
- hasUnconsumedMessages
  - activemq::core::ActiveMQSessionExecutor, 535
- have
  - gz\_state, 1977
  - inflate\_state, 2025
- havedict
  - inflate\_state, 2025
- HCRC
  - inflate.h, 4732
- hcrc
  - gz\_header\_s, 1975
- HCRC\_STATE
  - deflate.h, 4727
- HEAD
  - inflate.h, 4732
- head
  - inflate\_state, 2025
  - internal\_state, 2120
- HEADER\_ACK
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_CLIENT\_ID
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_CONSUMERPRIORITY
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_CONTENTLENGTH
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_CORRELATIONID
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_DESTINATION
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_DISPATCH\_ASYNC
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_EXCLUSIVE
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_EXPIRES
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_ID
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_JMSPRIORITY
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_LOGIN
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_MAXPENDINGMSGLIMIT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_MESSAGE
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_MESSAGEID
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_NOLOCAL
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_OLDSUBSCRIPTIONNAME
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_PASSWORD
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_PERSISTENT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_PREFETCHSIZE
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_RECEIPT\_REQUIRED
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_RECEIPTID
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_REDELIVERED
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- HEADER\_REDELIVERYCOUNT
  - activemq::wireformat::stomp::StompCommandConstants, 3631

Generated on Wed Jul 25 23:57:04 2012 for activemq-cpp-3.2.5 by Doxygen

- activemq::commands::ActiveMQTextMessage, 666
- ID\_ ACTIVEMQTOPIC
  - activemq::commands::ActiveMQTopic, 694
- ID\_ BROKERID
  - activemq::commands::BrokerId, 871
- ID\_ BROKERINFO
  - activemq::commands::BrokerInfo, 903
- ID\_ CONNECTIONCONTROL
  - activemq::commands::ConnectionControl, 1271
- ID\_ CONNECTIONERROR
  - activemq::commands::ConnectionError, 1299
- ID\_ CONNECTIONID
  - activemq::commands::ConnectionId, 1330
- ID\_ CONNECTIONINFO
  - activemq::commands::ConnectionInfo, 1360
- ID\_ CONSUMERCONTROL
  - activemq::commands::ConsumerControl, 1405
- ID\_ CONSUMERID
  - activemq::commands::ConsumerId, 1433
- ID\_ CONSUMERINFO
  - activemq::commands::ConsumerInfo, 1466
- ID\_ CONTROLCOMMAND
  - activemq::commands::ControlCommand, 1496
- ID\_ DATAARRAYRESPONSE
  - activemq::commands::DataArrayResponse, 1530
- ID\_ DATARESPONSE
  - activemq::commands::DataResponse, 1585
- ID\_ DESTINATIONINFO
  - activemq::commands::DestinationInfo, 1731
- ID\_ DISCOVERYEVENT
  - activemq::commands::DiscoveryEvent, 1761
- ID\_ EXCEPTIONRESPONSE
  - activemq::commands::ExceptionResponse, 1841
- ID\_ FLUSHCOMMAND
  - activemq::commands::FlushCommand, 1939
- ID\_ INTEGERRESPONSE
  - activemq::commands::IntegerResponse, 2093
- ID\_ JOURNALQUEUEACK
  - activemq::commands::JournalQueueAck, 2158
- ID\_ JOURNALTOPICACK
  - activemq::commands::JournalTopicAck, 2187
- ID\_ JOURNALTRACE
  - activemq::commands::JournalTrace, 2214
- ID\_ JOURNALTRANSACTION
  - activemq::commands::JournalTransaction, 2241
- ID\_ KEEPALIVEINFO
  - activemq::commands::KeepAliveInfo, 2268
- ID\_ LASTPARTIALCOMMAND
  - activemq::commands::LastPartialCommand, 2303
- ID\_ LOCALTRANSACTIONID
  - activemq::commands::LocalTransactionId, 2350
- ID\_ MESSAGE
  - activemq::commands::Message, 2532
- ID\_ MESSAGEACK
  - activemq::commands::MessageAck, 2564
- ID\_ MESSAGEDISPATCH
  - activemq::commands::MessageDispatch, 2598
- ID\_ MESSAGEDISPATCHNOTIFICATION
  - activemq::commands::MessageDispatchNotification, 2634
- ID\_ MESSAGEID
  - activemq::commands::MessageId, 2667
- ID\_ MESSAGEPULL
  - activemq::commands::MessagePull, 2743
- ID\_ NETWORKBRIDGEFILTER
  - activemq::commands::NetworkBridgeFilter, 2795
- ID\_ PARTIALCOMMAND
  - activemq::commands::PartialCommand, 2921
- ID\_ PRODUCERACK
  - activemq::commands::ProducerAck, 3039
- ID\_ PRODUCERID
  - activemq::commands::ProducerId, 3070
- ID\_ PRODUCERINFO
  - activemq::commands::ProducerInfo, 3100
- ID\_ REMOVEINFO
  - activemq::commands::RemoveInfo, 3197
- ID\_ REMOVESUBSCRIPTIONINFO
  - activemq::commands::RemoveSubscriptionInfo, 3226
- ID\_ REPLAYCOMMAND
  - activemq::commands::ReplayCommand, 3253
- ID\_ RESPONSE
  - activemq::commands::Response, 3288
- ID\_ SESSIONID
  - activemq::commands::SessionId, 3382
- ID\_ SESSIONINFO

- activemq::commands::SessionInfo, 3410
- ID\_SHUTDOWNINFO
  - activemq::commands::ShutdownInfo, 3472
- ID\_SUBSCRIPTIONINFO
  - activemq::commands::SubscriptionInfo, 3675
- ID\_TRANSACTIONID
  - activemq::commands::TransactionId, 3822
- ID\_TRANSACTIONINFO
  - activemq::commands::TransactionInfo, 3850
- ID\_WIREFORMATINFO
  - activemq::commands::WireFormatInfo, 3991
- ID\_XATransactionID
  - activemq::commands::XATransactionId, 4035
- IdGenerator
  - activemq::util::IdGenerator, 1989
- IllegalArgumentException
  - decaf::lang::exceptions::IllegalArgumentException, 1991, 1992
- IllegalMonitorStateException
  - decaf::lang::exceptions::IllegalMonitorStateException, 1994, 1995
- IllegalStateException
  - cms::IllegalStateException, 1997
  - decaf::lang::exceptions::IllegalStateException, 1998, 1999
- IllegalThreadStateException
  - decaf::lang::exceptions::IllegalThreadStateException, 2001, 2002
- impl
  - decaf::net::Socket, 3518
- implAccept
  - decaf::net::ServerSocket, 3358
- in
  - decaf::io::FileDescriptor, 1892
  - gz\_state, 1977
- InactivityMonitor
  - activemq::transport::inactivity::InactivityMonitor, 2005
- increaseUsage
  - activemq::util::MemoryUsage, 2514
  - activemq::util::Usage, 3965
- incrementAndGet
  - decaf::util::concurrent::atomic::AtomicInteger, 744
- indexOf
  - decaf::util::List, 2339
  - decaf::util::StlList, 3597
- IndexOutOfBounds Exception
  - decaf::lang::exceptions::IndexOutOfBounds Exception, 2008, 2009
- INDIVIDUAL\_ACKNOWLEDGE
  - cms::Session, 3368
- Inet4Address
  - decaf::net::Inet4Address, 2012
- Inet6Address
  - decaf::net::Inet6Address, 2015
- InetAddress
  - decaf::net::Inet4Address, 2014
  - decaf::net::Inet6Address, 2015
  - decaf::net::InetAddress, 2018
- InetSocketAddress
  - decaf::net::InetSocketAddress, 2023
- inffast.h
  - OF, 4730
- inflate
  - decaf::util::zip::Inflater, 2030, 2031
- inflate.h
  - BAD, 4733
  - CHECK, 4733
  - CODELENS, 4733
  - COMMENT, 4732
  - COPY, 4733
  - COPY\_, 4733
  - DICT, 4732
  - DICTID, 4732
  - DIST, 4733
  - DISTEXT, 4733
  - DONE, 4733
  - EXLEN, 4732
  - EXTRA, 4732
  - FLAGS, 4732
  - GUNZIP, 4732
  - HCRC, 4732
  - HEAD, 4732
  - inflate\_mode, 4732
  - LEN, 4733
  - LEN\_, 4733
  - LENEXT, 4733
  - LENGTH, 4733
  - LENLENS, 4733
  - LIT, 4733
  - MATCH, 4733
  - MEM, 4733
  - NAME, 4732
  - OS, 4732
  - STORED, 4733
  - SYNC, 4733
  - TABLE, 4733
  - TIME, 4732
  - TYPE, 4732
  - TYPEDO, 4732
- inflate\_mode
  - inflate.h, 4732
- inflate\_state, 2024

- back, 2025
- bits, 2025
- check, 2025
- codes, 2025
- distbits, 2025
- distcode, 2025
- dmax, 2025
- extra, 2025
- flags, 2025
- have, 2025
- havedict, 2025
- head, 2025
- hold, 2025
- last, 2025
- lenbits, 2025
- lencode, 2025
- length, 2025
- lens, 2025
- mode, 2025
- ncode, 2025
- ndist, 2025
- next, 2025
- nlen, 2025
- offset, 2025
- sane, 2025
- total, 2025
- was, 2025
- wbits, 2025
- whave, 2025
- window, 2025
- wnext, 2025
- work, 2025
- wrap, 2025
- wsiz, 2025
- inflateBackInit
  - zlib.h, 4741
- inflateInit
  - zlib.h, 4741
- inflateInit2
  - zlib.h, 4742
- Inflater
  - decaf::util::zip::Inflater, 2029
- inflater
  - decaf::util::zip::InflaterInputStream, 2041
- InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 2037, 2038
- INFO
  - decaf::util::logging::Level, 2336
- Info
  - decaf::util::logging, 176
- info
  - decaf::util::logging::Logger, 2393
  - decaf::util::logging::SimpleLogger, 3502
- inftrees.h
  - CODES, 4734
  - codetype, 4734
  - DISTS, 4734
  - ENOUGH, 4734
  - ENOUGH\_DISTS, 4734
  - ENOUGH\_LENS, 4734
  - LENS, 4734
  - OF, 4734
- INHERIT
  - decaf::util::logging::Level, 2336
- init
  - activemq::cmsutil::CmsAccessor, 1156
  - activemq::cmsutil::CmsDestinationAccessor, 1158
  - activemq::cmsutil::CmsTemplate, 1176
  - activemq::cmsutil::DestinationResolver, 1756
  - activemq::cmsutil::DynamicDestinationResolver, 1822
- INIT\_STATE
  - deflate.h, 4727
- initCause
  - decaf::lang::Exception, 1836
  - decaf::lang::Throwable, 3786
- initializeLibrary
  - activemq::library::ActiveMQCPP, 320, 321
- initializeNetworking
  - decaf::internal::net::Network, 2791
- initializeRuntime
  - decaf::lang::Runtime, 3326
- initSocketImpl
  - decaf::net::Socket, 3513
- inProgressClearRequired
  - activemq::core::ActiveMQConsumer, 316
- InputStream
  - decaf::io::InputStream, 2045
- inputStream
  - decaf::io::FilterInputStream, 1899
- InputStreamReader
  - decaf::io::InputStreamReader, 2054
- inReceive
  - activemq::wireformat::openwire::OpenWireFormat, 2892
  - activemq::wireformat::stomp::StompWireFormat, 3644
  - activemq::wireformat::WireFormat, 3978
- ins\_h
  - internal\_state, 2120
- insert
  - decaf::internal::util::TimerTaskHeap, 3805
- IntArrayBuffer
  - decaf::internal::nio::IntArrayBuffer, 2059, 2060

- intBitsToFloat
  - decaf::lang::Float, 1909
- IntBuffer
  - decaf::nio::IntBuffer, 2068
- Integer
  - decaf::lang::Integer, 2079
- INTEGER\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- IntegerResponse
  - activemq::commands::IntegerResponse, 2092
- IntegerResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2111
  - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2099
  - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2103
  - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2107
  - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2115
  - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2095
- internal\_state, 2118
  - bi\_buf, 2120
  - bi\_valid, 2120
  - bl\_count, 2120
  - bl\_desc, 2120
  - bl\_tree, 2120
  - block\_start, 2120
  - d\_buf, 2120
  - d\_desc, 2120
  - depth, 2120
  - dummy, 2120
  - dyn\_dtree, 2120
  - dyn\_ltree, 2120
  - good\_match, 2120
  - gzhead, 2120
  - gzindex, 2120
  - hash\_bits, 2120
  - hash\_mask, 2120
  - hash\_shift, 2120
  - hash\_size, 2120
  - head, 2120
  - heap, 2120
  - heap\_len, 2120
  - heap\_max, 2120
  - high\_water, 2120
  - ins\_h, 2120
  - l\_buf, 2120
  - l\_desc, 2120
  - last\_eob\_len, 2120
  - last\_flush, 2120
  - last\_lit, 2120
  - level, 2120
  - lit\_bufsize, 2120
  - lookahead, 2120
  - match\_available, 2120
  - match\_length, 2120
  - match\_start, 2120
  - matches, 2120
  - max\_chain\_length, 2120
  - max\_lazy\_match, 2120
  - method, 2120
  - nice\_match, 2120
  - opt\_len, 2120
  - pending, 2120
  - pending\_buf, 2120
  - pending\_buf\_size, 2120
  - pending\_out, 2120
  - prev, 2120
  - prev\_length, 2120
  - prev\_match, 2120
  - static\_len, 2120
  - status, 2120
  - strategy, 2120
  - strm, 2120
  - strstart, 2120
  - w\_bits, 2120
  - w\_mask, 2120
  - w\_size, 2120
  - window, 2120
  - window\_size, 2120
  - wrap, 2120
- InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2122
- InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2124, 2125
- InterruptedIOException
  - decaf::io::InterruptedIOException, 2127, 2128
- intf
  - zconf.h, 4738
- intValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
  - decaf::lang::Byte, 964
  - decaf::lang::Character, 1103
  - decaf::lang::Double, 1793
  - decaf::lang::Float, 1909
  - decaf::lang::Integer, 2082
  - decaf::lang::Long, 2425
  - decaf::lang::Number, 2836
  - decaf::lang::Short, 3444

- decaf::util::concurrent::atomic::AtomicIntegerBrowser
  - 745
- decaf::util::logging::Level, 2334
- InvalidClientIdException
  - cms::InvalidClientIdException, 2130
- InvalidDestinationException
  - cms::InvalidDestinationException, 2131
- InvalidKeyException
  - decaf::security::InvalidKeyException, 2132, 2133
- InvalidMarkException
  - decaf::nio::InvalidMarkException, 2135, 2136
- InvalidSelectorException
  - cms::InvalidSelectorException, 2138
- InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2139, 2140
- IOException
  - decaf::io::IOException, 2142, 2143
- IOTransport
  - activemq::transport::IOTransport, 2146
- IPos
  - deflate.h, 4727
- isAbsolute
  - decaf::internal::net::URIType, 3955
  - decaf::net::URI, 3928
- isAdvisory
  - activemq::commands::ActiveMQDestination, 327
- isAlive
  - decaf::lang::Thread, 3770
- isAlwaysSyncSend
  - activemq::core::ActiveMQConnection, 284
  - activemq::core::ActiveMQConnectionFactory, 298
- isAnyLocalAddress
  - decaf::net::Inet4Address, 2012
  - decaf::net::InetAddress, 2020
- isAutoAcknowledge
  - activemq::core::ActiveMQSession, 528
- isBackup
  - activemq::transport::failover::FailoverTransport, 1878
- isBound
  - decaf::net::ServerSocket, 3358
  - decaf::net::Socket, 3513
- isBrokerInfo
  - activemq::commands::BaseCommand, 760
  - activemq::commands::BrokerInfo, 900
  - activemq::commands::Command, 1195
- isBrokerMasterConnector
  - activemq::commands::ConnectionInfo, 1358
- activemq::commands::ConsumerInfo, 1463
- isBusy
  - decaf::util::concurrent::PooledThread, 2969
- isCacheEnabled
  - activemq::commands::WireFormatInfo, 3987
  - activemq::wireformat::openwire::OpenWireFormat, 2892
- isCancelled
  - decaf::util::concurrent::Future, 1968
- isClientAcknowledge
  - activemq::core::ActiveMQSession, 528
- isClientMaster
  - activemq::commands::ConnectionInfo, 1358
- isClose
  - activemq::commands::ConnectionControl, 1270
  - activemq::commands::ConsumerControl, 1404
- isClosed
  - activemq::core::ActiveMQConnection, 284
  - activemq::core::ActiveMQConsumer, 316
  - activemq::core::ActiveMQProducer, 474
  - activemq::core::MessageDispatchChannel, 2601
  - activemq::transport::failover::BackupTransport, 753
  - activemq::transport::failover::FailoverTransport, 1878
  - activemq::transport::IOTransport, 2147
  - activemq::transport::mock::MockTransport, 2772
  - activemq::transport::tcp::TcpTransport, 3754
  - activemq::transport::Transport, 3884
  - activemq::transport::TransportFilter, 3894
  - decaf::internal::net::tcp::TcpSocket, 3744
  - decaf::io::FilterInputStream, 1897
  - decaf::io::FilterOutputStream, 1902
  - decaf::net::ServerSocket, 3358
  - decaf::net::Socket, 3513
- isComposite
  - activemq::commands::ActiveMQDestination, 328
- isCompressed
  - activemq::commands::Message, 2526
- isConnected
  - activemq::transport::failover::FailoverTransport, 1878
  - activemq::transport::IOTransport, 2148
  - activemq::transport::mock::MockTransport, 2772



- activemq::transport::tcp::TcpTransport, 3754
- activemq::transport::Transport, 3885
- activemq::transport::TransportFilter, 3894
- decaf::internal::net::tcp::TcpSocket, 3744
- decaf::net::Socket, 3514
- isConnectionAdvisory
  - activemq::commands::ActiveMQDestination, 328
- isConnectionInfo
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1195
  - activemq::commands::ConnectionInfo, 1358
- isConnectionInterruptProcessingComplete
  - activemq::state::ConnectionState, 1391
- isConsumerAdvisory
  - activemq::commands::ActiveMQDestination, 328
- isConsumerInfo
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1195
  - activemq::commands::ConsumerInfo, 1463
- isDeletedByBroker
  - activemq::commands::ActiveMQBlobMessage, 207
- isDigit
  - decaf::lang::Character, 1104
- isDispatchAsync
  - activemq::commands::ConsumerInfo, 1463
  - activemq::commands::ProducerInfo, 3098
  - activemq::core::ActiveMQConnection, 284
  - activemq::core::ActiveMQConnectionFactory, 298
- isDone
  - decaf::util::concurrent::Future, 1968
  - decaf::util::zip::DeflaterOutputStream, 1719
- isDroppable
  - activemq::commands::Message, 2526
- isDuplexConnection
  - activemq::commands::BrokerInfo, 900
- isDupsOkAcknowledge
  - activemq::core::ActiveMQSession, 528
- isEmpty
  - activemq::core::ActiveMQSessionExecutor, 535
  - activemq::core::MessageDispatchChannel, 2601
  - activemq::util::ActiveMQProperties, 480
  - cms::CMSProperties, 1167
  - decaf::internal::util::TimerTaskHeap, 3805
  - decaf::lang::String, 3666
  - decaf::util::AbstractCollection, 185
- decaf::util::Collection, 1189
- decaf::util::concurrent::ConcurrentStlMap, 1240
- decaf::util::concurrent::SynchronousQueue, 3720
- decaf::util::Map, 2465
- decaf::util::Properties, 3129
- decaf::util::StlList, 3597
- decaf::util::StlMap, 3609
- decaf::util::StlSet, 3627
- isEnabled
  - activemq::transport::failover::BackupTransportPool, 756
- isExclusive
  - activemq::commands::ActiveMQDestination, 328
  - activemq::commands::ConsumerInfo, 1464
- isExit
  - activemq::commands::ConnectionControl, 1270
- isExpired
  - activemq::commands::Message, 2526
- isExplicitQosEnabled
  - activemq::cmsutil::CmsTemplate, 1176
- isFailOnClose
  - activemq::transport::mock::MockTransport, 2772
- isFailOnKeepAliveSends
  - activemq::transport::mock::MockTransport, 2773
- isFailOnReceiveMessage
  - activemq::transport::mock::MockTransport, 2773
- isFailOnSendMessage
  - activemq::transport::mock::MockTransport, 2773
- isFailOnStart
  - activemq::transport::mock::MockTransport, 2773
- isFailOnStop
  - activemq::transport::mock::MockTransport, 2773
- isFair
  - decaf::util::concurrent::locks::ReentrantLock, 3184
  - decaf::util::concurrent::Semaphore, 3346
- isFaultTolerant
  - activemq::commands::ConnectionControl, 1270
  - activemq::commands::ConnectionInfo, 1358
  - activemq::transport::failover::FailoverTransport, 1878
  - activemq::transport::IOTransport, 2148

- activemq::transport::mock::MockTransport, 2773
- activemq::transport::tcp::TcpTransport, 3754
- activemq::transport::Transport, 3885
- activemq::transport::TransportFilter, 3894
- isFaultTolerantConfiguration
  - activemq::commands::BrokerInfo, 901
- isFlush
  - activemq::commands::ConsumerControl, 1404
- isFull
  - activemq::util::MemoryUsage, 2514
  - activemq::util::Usage, 3965
- isHeldByCurrentThread
  - decaf::util::concurrent::locks::ReentrantLock, 3184
- isIndividualAcknowledge
  - activemq::core::ActiveMQSession, 528
- isInfinite
  - decaf::lang::Double, 1793
  - decaf::lang::Float, 1909, 1910
- isInitialized
  - activemq::transport::failover::FailoverTransport, 1878
- isInputShutdown
  - decaf::net::Socket, 3514
- isInTransaction
  - activemq::core::ActiveMQTransactionContext, 721
- isISOCtrl
  - decaf::lang::Character, 1104
- isKeepAliveInfo
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1195
  - activemq::commands::KeepAliveInfo, 2268
- isKeepAliveResponseRequired
  - activemq::transport::inactivity::InactivityMonitor, 2005
- isLetter
  - decaf::lang::Character, 1104
- isLetterOrDigit
  - decaf::lang::Character, 1104
- isLinkLocalAddress
  - decaf::net::Inet4Address, 2012
  - decaf::net::InetAddress, 2020
- isLocked
  - decaf::util::concurrent::Lock, 2376
  - decaf::util::concurrent::locks::ReentrantLock, 3184
- isLoggable
  - decaf::util::logging::Filter, 1893
  - decaf::util::logging::Handler, 1980
  - decaf::util::logging::Logger, 2393
- decaf::util::logging::StreamHandler, 3651
- isLoopbackAddress
  - decaf::net::Inet4Address, 2012
  - decaf::net::InetAddress, 2020
- isLowerCase
  - decaf::lang::Character, 1104
- isManageable
  - activemq::commands::ConnectionInfo, 1359
- isMarshalAware
  - activemq::commands::ActiveMQMapMessage, 368
  - activemq::commands::BaseDataStructure, 832
  - activemq::commands::Message, 2526
  - activemq::commands::WireFormatInfo, 3987
  - activemq::wireformat::MarshalAware, 2485
- isMasterBroker
  - activemq::commands::BrokerInfo, 901
- isMCGlobal
  - decaf::net::Inet4Address, 2012
  - decaf::net::InetAddress, 2020
- isMCLinkLocal
  - decaf::net::Inet4Address, 2013
  - decaf::net::InetAddress, 2021
- isMCNodeLocal
  - decaf::net::Inet4Address, 2013
  - decaf::net::InetAddress, 2021
- isMCOrgLocal
  - decaf::net::Inet4Address, 2013
  - decaf::net::InetAddress, 2021
- isMCSiteLocal
  - decaf::net::Inet4Address, 2013
  - decaf::net::InetAddress, 2021
- isMessage
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1195
  - activemq::commands::Message, 2527
- isMessageAck
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1195
  - activemq::commands::MessageAck, 2562
- isMessageDispatch
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1196
  - activemq::commands::MessageDispatch, 2596
- isMessageDispatchNotification
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1196
  - activemq::commands::MessageDispatchNotification, 2632
- isMessageIdEnabled

- activemq::cmsutil::CmsTemplate, 1176
- isMessageTimestampEnabled
  - activemq::cmsutil::CmsTemplate, 1176
- isMulticastAddress
  - decaf::net::Inet4Address, 2013
  - decaf::net::InetAddress, 2021
- isNaN
  - decaf::lang::Double, 1794
  - decaf::lang::Float, 1910
- isNetworkConnection
  - activemq::commands::BrokerInfo, 901
- isNetworkSubscription
  - activemq::commands::ConsumerInfo, 1464
- isNoLocal
  - activemq::cmsutil::CmsTemplate, 1176
  - activemq::commands::ConsumerInfo, 1464
- isNoRangeAcks
  - activemq::commands::ConsumerInfo, 1464
- isOpaque
  - decaf::internal::net::URIType, 3956
  - decaf::net::URI, 3928
- isOptimizedAcknowledge
  - activemq::commands::ConsumerInfo, 1464
- isOrdered
  - activemq::commands::ActiveMQDestination, 328
- isOutputShutdown
  - decaf::net::Socket, 3514
- isPending
  - activemq::threads::CompositeTask, 1221
  - activemq::transport::failover::BackupTransportPool, 756
  - activemq::transport::failover::CloseTransportsTask, 1151
  - activemq::transport::failover::FailoverTransport, 1879
- isPersistent
  - activemq::commands::Message, 2527
- isPrepared
  - activemq::state::TransactionState, 3876
- isProducerAck
  - activemq::commands::BaseCommand, 761
  - activemq::commands::Command, 1196
  - activemq::commands::ProducerAck, 3038
- isProducerAdvisory
  - activemq::commands::ActiveMQDestination, 328
- isProducerInfo
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1196
  - activemq::commands::ProducerInfo, 3098
- isPubSubDomain
  - activemq::cmsutil::CmsDestinationAccessor, 1158
- isQueue
  - activemq::commands::ActiveMQDestination, 329
- isRandomize
  - activemq::transport::failover::FailoverTransport, 1879
  - activemq::transport::failover::URIPool, 3943
- isReadOnly
  - decaf::internal::nio::ByteBuffer, 1012
  - decaf::internal::nio::CharArrayBuffer, 1116
  - decaf::internal::nio::DoubleArrayBuffer, 1806
  - decaf::internal::nio::FloatArrayBuffer, 1922
  - decaf::internal::nio::IntArrayBuffer, 2063
  - decaf::internal::nio::LongArrayBuffer, 2441
  - decaf::internal::nio::ShortArrayBuffer, 3456
  - decaf::nio::Buffer, 931
  - decaf::nio::ByteBuffer, 1046
- isReadOnlyBody
  - activemq::commands::Message, 2527
- isReadOnlyProperties
  - activemq::commands::Message, 2527
- isRebalanceConnection
  - activemq::commands::ConnectionControl, 1270
- isRecievedByDFBridge
  - activemq::commands::Message, 2527
- isRemoveInfo
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1196
  - activemq::commands::RemoveInfo, 3196
- isRemoveSubscriptionInfo
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1196
  - activemq::commands::RemoveSubscriptionInfo, 3224
- isResponse
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1196
  - activemq::commands::Response, 3287
- isResponseRequired
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1196
- isRestoreConsumers
  - activemq::state::ConnectionStateTracker, 1394
- isRestoreProducers
  - activemq::state::ConnectionStateTracker, 1394
- isRestoreSessions
  - activemq::state::ConnectionStateTracker, 1394

- isRestoreTransaction
  - activemq::state::ConnectionStateTracker, 1394
- isResume
  - activemq::commands::ConnectionControl, 1270
- isRetroactive
  - activemq::commands::ConsumerInfo, 1464
- isRunning
  - activemq::core::ActiveMQSessionExecutor, 535
  - activemq::core::MessageDispatchChannel, 2602
- isScheduled
  - decaf::util::TimerTask, 3802
- isServerAuthority
  - decaf::internal::net::URIType, 3956
- isShutdownInfo
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1197
  - activemq::commands::ShutdownInfo, 3472
- isSiteLocalAddress
  - decaf::net::Inet4Address, 2014
  - decaf::net::InetAddress, 2022
- isSizePrefixDisabled
  - activemq::commands::WireFormatInfo, 3988
  - activemq::wireformat::openwire::OpenWireFormat, 2892
- isSlaveBroker
  - activemq::commands::BrokerInfo, 901
- isStackTraceEnabled
  - activemq::commands::WireFormatInfo, 3988
  - activemq::wireformat::openwire::OpenWireFormat, 2893
- isStart
  - activemq::commands::ConsumerControl, 1404
- isStarted
  - activemq::core::ActiveMQConnection, 284
  - activemq::core::ActiveMQSession, 528
- isStop
  - activemq::commands::ConsumerControl, 1404
- isSuspend
  - activemq::commands::ConnectionControl, 1270
- isSynchronizationRegistered
  - activemq::core::ActiveMQConsumer, 316
- isTcpNoDelayEnabled
  - activemq::commands::WireFormatInfo, 3988
- activemq::wireformat::openwire::OpenWireFormat, 2893
- isTemporary
  - activemq::commands::ActiveMQDestination, 329
- isTightEncodingEnabled
  - activemq::commands::WireFormatInfo, 3988
  - activemq::wireformat::openwire::OpenWireFormat, 2893
- isTopic
  - activemq::commands::ActiveMQDestination, 329
- isTrackMessages
  - activemq::state::ConnectionStateTracker, 1394
  - activemq::transport::failover::FailoverTransport, 1879
- isTrackTransactionProducers
  - activemq::state::ConnectionStateTracker, 1394
  - activemq::transport::failover::FailoverTransport, 1879
- isTrackTransactions
  - activemq::state::ConnectionStateTracker, 1394
- isTransacted
  - activemq::cmsutil::PooledSession, 2965
  - activemq::core::ActiveMQSession, 529
  - cms::Session, 3375
- isTransactionInfo
  - activemq::commands::BaseCommand, 762
  - activemq::commands::Command, 1197
  - activemq::commands::TransactionInfo, 3849
- isTransportFailed
  - activemq::core::ActiveMQConnection, 284
- isUpperCase
  - decaf::lang::Character, 1104
- isUseAsyncSend
  - activemq::core::ActiveMQConnection, 285
  - activemq::core::ActiveMQConnectionFactory, 299
- isUseCollisionAvoidance
  - activemq::core::policies::DefaultRedeliveryPolicy, 1679
  - activemq::core::RedeliveryPolicy, 3180
- isUseCompression
  - activemq::core::ActiveMQConnection, 285
  - activemq::core::ActiveMQConnectionFactory, 299
- isUseExponentialBackOff
  - activemq::core::policies::DefaultRedeliveryPolicy, 1679

- activemq::core::RedeliveryPolicy, 3180
- activemq::transport::failover::FailoverTransport, 1879
- isValid
  - activemq::commands::WireFormatInfo, 3988
  - decaf::internal::net::URIType, 3956
- isValidDomainName
  - decaf::internal::net::URIHelper, 3937
- isValidHexChar
  - decaf::internal::net::URIHelper, 3937
- isValidHost
  - decaf::internal::net::URIHelper, 3937
- isValidIP4Word
  - decaf::internal::net::URIHelper, 3937
- isValidIP6Address
  - decaf::internal::net::URIHelper, 3938
- isValidIPv4Address
  - decaf::internal::net::URIHelper, 3938
- isWaitingForResponse
  - activemq::state::Tracked, 3818
- isWhitespace
  - decaf::lang::Character, 1104
- isWildcard
  - activemq::commands::ActiveMQDestination, 329
- isWireFormatInfo
  - activemq::commands::BaseCommand, 763
  - activemq::commands::Command, 1197
  - activemq::commands::WireFormatInfo, 3988
- itemExists
  - activemq::commands::ActiveMQMapMessage, 368
  - cms::MapMessage, 2478
- iterate
  - activemq::core::ActiveMQConsumer, 316
  - activemq::core::ActiveMQSessionExecutor, 535
  - activemq::threads::CompositeTaskRunner, 1224
  - activemq::threads::Task, 3734
  - activemq::transport::failover::BackupTransportPool, 757
  - activemq::transport::failover::CloseTransportsTask, 1151
  - activemq::transport::failover::FailoverTransport, 1879
- iterator
  - decaf::lang::Iterable, 2152
  - decaf::util::concurrent::SynchronousQueue, 3721
  - decaf::util::PriorityQueue, 3032
  - decaf::util::StlList, 3598
- decaf::util::StlQueue, 3618
- decaf::util::StlSet, 3627
- join
  - decaf::lang::Thread, 3770, 3771
- JournalQueueAck
  - activemq::commands::JournalQueueAck, 2157
- JournalQueueAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalQueueAck, 2180
  - activemq::wireformat::openwire::marshal::v2::JournalQueueAck, 2164
  - activemq::wireformat::openwire::marshal::v3::JournalQueueAck, 2172
  - activemq::wireformat::openwire::marshal::v4::JournalQueueAck, 2176
  - activemq::wireformat::openwire::marshal::v5::JournalQueueAck, 2168
  - activemq::wireformat::openwire::marshal::v6::JournalQueueAck, 2160
- JournalTopicAck
  - activemq::commands::JournalTopicAck, 2184
- JournalTopicAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTopicAck, 2209
  - activemq::wireformat::openwire::marshal::v2::JournalTopicAck, 2193
  - activemq::wireformat::openwire::marshal::v3::JournalTopicAck, 2197
  - activemq::wireformat::openwire::marshal::v4::JournalTopicAck, 2205
  - activemq::wireformat::openwire::marshal::v5::JournalTopicAck, 2189
  - activemq::wireformat::openwire::marshal::v6::JournalTopicAck, 2201
- JournalTrace
  - activemq::commands::JournalTrace, 2213
- JournalTraceMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTraceMa, 2216
  - activemq::wireformat::openwire::marshal::v2::JournalTraceMa, 2220
  - activemq::wireformat::openwire::marshal::v3::JournalTraceMa, 2228
  - activemq::wireformat::openwire::marshal::v4::JournalTraceMa, 2236
  - activemq::wireformat::openwire::marshal::v5::JournalTraceMa, 2224
  - activemq::wireformat::openwire::marshal::v6::JournalTraceMa, 2224
- JournalTransaction

- activemq::commands::JournalTransaction, 2240
- JournalTransactionMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2263
  - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2247
  - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2251
  - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2259
  - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2255
  - activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2243
- KeepAliveInfo
  - activemq::commands::KeepAliveInfo, 2267
- KeepAliveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2290
  - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2274
  - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2278
  - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2282
  - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2286
  - activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2270
- KeyException
  - decaf::security::KeyException, 2295, 2296
- KeyManagementException
  - decaf::security::KeyManagementException, 2298, 2299
- keySet
  - decaf::util::concurrent::ConcurrentStlMap, 1240
  - decaf::util::Map, 2466
  - decaf::util::StlMap, 3609
- l\_buf
  - internal\_state, 2120
- L\_CODES
  - deflate.h, 4727
- l\_desc
  - internal\_state, 2120
- last
  - inflate\_state, 2025
- last\_eob\_len
  - internal\_state, 2120
- last\_flush
  - internal\_state, 2120
- last\_lit
  - internal\_state, 2120
- lastDeliveredSequenceId
  - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2263
- lastIndexOf
  - decaf::util::StlList, 3598
- lastPartialCommand
  - activemq::commands::MessageAck, 2564
- LastPartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2325
  - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2313
  - activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2309
  - activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2311
  - activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2317
  - activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2305
- LEN
  - deflate.h, 4727
- len
  - ct\_data\_s, 1527
- LEN\_
  - inflate.h, 4733
- lenbits
  - inflate\_state, 2025
- lencode
  - inflate\_state, 2025
- LENEXT
  - inflate.h, 4733
- LENGTH
  - inflate.h, 4733
- length
  - decaf::internal::nio::CharArrayBuffer, 1118
  - decaf::lang::ArrayPointer, 733
  - decaf::lang::CharSequence, 1136
  - decaf::lang::String, 3667
  - decaf::nio::CharBuffer, 1128
  - decaf::util::zip::InflaterInputStream, 2041
  - inflate\_state, 2025
- LENGTH\_CODES
  - deflate.h, 4727

- LENLENS
  - inflate.h, 4733
- LENS
  - inftrees.h, 4734
- lens
  - inflate\_state, 2025
- Less
  - decaf::util::comparators::Less, 2328
- Level
  - decaf::util::logging::Level, 2333
- level
  - gz\_state, 1977
  - internal\_state, 2120
- Levels
  - decaf::util::logging, 176
- limit
  - decaf::nio::Buffer, 931, 932
- LineNumber
  - activemq::commands::BrokerError::StackTraceElement, 3578
- List
  - decaf::util::List, 2338
- LIST\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- listen
  - decaf::internal::net::tcp::TcpSocket, 3744
  - decaf::net::SocketImpl, 3534
- listener
  - activemq::transport::TransportFilter, 3898
- listIterator
  - decaf::util::List, 2340, 2341
  - decaf::util::StlList, 3598, 3599
- listValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
- LIT
  - inflate.h, 4733
- lit\_bufsize
  - internal\_state, 2120
- LITERALS
  - deflate.h, 4727
- load
  - decaf::util::Properties, 3129, 3131
- local
  - gzguts.h, 4729
- localPort
  - decaf::net::SocketImpl, 3536
- LocalTransactionId
  - activemq::commands::LocalTransactionId, 2348
- LocalTransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2372
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2356
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2360
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2368
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2364
- activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2352
- Lock
  - decaf::util::concurrent::Lock, 2375
- lock
  - activemq::core::MessageDispatchChannel, 2602
  - decaf::internal::util::concurrent::MutexImpl, 2788
  - decaf::internal::util::concurrent::SynchronizableImpl, 3712
  - decaf::io::InputStream, 2046
  - decaf::io::OutputStream, 2910
  - decaf::util::AbstractCollection, 186
  - decaf::util::concurrent::ConcurrentStlMap, 1241
  - decaf::util::concurrent::Lock, 2376
  - decaf::util::concurrent::locks::Lock, 2378
  - decaf::util::concurrent::locks::ReentrantLock, 3184
  - decaf::util::concurrent::Mutex, 2783
  - decaf::util::concurrent::Synchronizable, 3701
  - decaf::util::StlMap, 3609
  - decaf::util::StlQueue, 3618
- lock\_count
  - decaf::util::concurrent::MutexHandle, 2787
- lock\_owner
  - decaf::util::concurrent::MutexHandle, 2787
- lockInterruptibly
  - decaf::util::concurrent::locks::Lock, 2379
  - decaf::util::concurrent::locks::ReentrantLock, 3185
- log
  - decaf::util::logging::Logger, 2394
  - decaf::util::logging::LogWriter, 2418, 2419
  - decaf::util::logging::SimpleLogger, 3502
- LOGDECAF\_DEBUG
  - LoggerDefines.h, 4933
- LOGDECAF\_DEBUG\_1
  - LoggerDefines.h, 4933
- LOGDECAF\_DECLARE
  - LoggerDefines.h, 4933
- LOGDECAF\_DECLARE\_LOCAL
  - LoggerDefines.h, 4933
- LOGDECAF\_ERROR
  - LoggerDefines.h, 4933

- LoggerDefines.h, 4934
- LOGDECAF\_FATAL
  - LoggerDefines.h, 4934
- LOGDECAF\_INFO
  - LoggerDefines.h, 4934
- LOGDECAF\_INITIALIZE
  - LoggerDefines.h, 4934
- LOGDECAF\_WARN
  - LoggerDefines.h, 4934
- Logger
  - decaf::util::logging::Logger, 2389
- LoggerDefines.h
  - LOGDECAF\_DEBUG, 4933
  - LOGDECAF\_DEBUG\_1, 4933
  - LOGDECAF\_DECLARE, 4933
  - LOGDECAF\_DECLARE\_LOCAL, 4934
  - LOGDECAF\_ERROR, 4934
  - LOGDECAF\_FATAL, 4934
  - LOGDECAF\_INFO, 4934
  - LOGDECAF\_INITIALIZE, 4934
  - LOGDECAF\_WARN, 4934
- LoggerHierarchy
  - decaf::util::logging::LoggerHierarchy, 2398
- LoggingInputStream
  - activemq::io::LoggingInputStream, 2399
- LoggingOutputStream
  - activemq::io::LoggingOutputStream, 2401
- LoggingTransport
  - activemq::transport::logging::LoggingTransport, 2403
- LogManager
  - decaf::util::logging::LogManager, 2408
- LogRecord
  - decaf::util::logging::LogRecord, 2414
- LogWriter
  - decaf::util::logging::LogWriter, 2418
- Long
  - decaf::lang::Long, 2422
- LONG\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- LongArrayBuffer
  - decaf::internal::nio::LongArrayBuffer, 2437, 2438
- longBitsToDouble
  - decaf::lang::Double, 1794
- LongBuffer
  - decaf::nio::LongBuffer, 2446
- LongSequenceGenerator
  - activemq::util::LongSequenceGenerator, 2455
- longValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3009
  - decaf::lang::Byte, 964
  - decaf::lang::Character, 1104
  - decaf::lang::Double, 1794
  - decaf::lang::Float, 1910
  - decaf::lang::Integer, 2082
  - decaf::lang::Long, 2425
  - decaf::lang::Number, 2836
  - decaf::lang::Short, 3444
  - decaf::util::concurrent::atomic::AtomicInteger, 745
- LOOK
  - gzguts.h, 4729
- lookahead
  - internal\_state, 2120
- loopbackBytes
  - decaf::net::InetAddress, 2022
- looseMarshal
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshal, 813
  - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1623
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshal, 214
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshal, 254
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestination, 338
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshal, 379
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 406
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshal, 451
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshal, 496
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshal, 557
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplate, 585
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplate, 614
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplate, 647
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshal, 676
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshal, 704
  - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshal, 780
  - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshal, 881
  - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshal, 913
  - activemq::wireformat::openwire::marshal::v1::ConnectionContext, 1281



activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2817
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFactory	1313
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2943
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	1344
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	1374
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	3061
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	1419
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	3093
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1448
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3110
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1481
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3211
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	1510
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	3228
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	1544
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	3259
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	1607
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3316
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	1745
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3404
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	1779
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3420
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	1863
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3482
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	1957
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3681
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	2111
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3828
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	2180
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3856
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	2209
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	4009
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	2232
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	4049
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	2263
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	222
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	2290
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	270
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	2325
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	350
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	2372
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	391
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2582
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	418
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	2623
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	463
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2652
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	508
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	2689
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	569
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2714
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	597
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2761
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	626

activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	2356	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	2356
655		activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2570
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2570	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2607
688		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	2607	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
716		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2640	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
801		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2669	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
893		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2704	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
925		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2745	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1293		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2797	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1301		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2927	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1332		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	3041	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1362		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	3073	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1407		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	3106	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1436		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3199	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1469		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ContactGroupAndIdMarshaller	3236	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1498		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3263	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1532		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3301	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1595		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3384	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1733		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	3428	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1767		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3478	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1847		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3697	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
1945		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3832	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2099		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3872	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2164		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	4001	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2193		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	4041	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2216		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	210	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2247		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	250	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2274		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	334	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669
2313		activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2669

activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	2103
375	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2172
402	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2197
447	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	2220
492	
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceMessageMarshaller	2251
553	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	2278
581	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2309
610	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2360
639	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2574
668	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2611
696	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	2644
766	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2681
873	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2699
905	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2753
1273	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2809
1305	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2935
1336	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	3049
1366	
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	3081
1411	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	3118
1440	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	3207
1473	
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	3232
1502	
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	3267
1536	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3311
1599	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	3400
1737	
activemq::wireformat::openwire::marshal::v3::DiscardEventMarshaller	3424
1771	
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	3490
1851	
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3677
1949	

activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	1506
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3836
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3860
activemq::wireformat::openwire::marshal::v3::WireFormatInwardMarshaller	1540
activemq::wireformat::openwire::marshal::v3::WireFormatInwardMarshaller	4013
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1603
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	4053
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	1741
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	218
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	1775
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	258
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1859
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	342
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	1953
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	383
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2107
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	410
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2176
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	455
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2205
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	500
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	2228
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	561
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	2259
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	589
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2282
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	618
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2321
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	643
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2368
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	672
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2578
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	700
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	2619
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	773
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2648
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	877
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	909
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1277
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1309
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1340
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1370
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1415
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1444
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1477
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1506
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1540
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1603
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1741
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1775
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1859
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	1953
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2107
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2176
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2205
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2228
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2259
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2282
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2321
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2368
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2578
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2619
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2648
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2673
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2709
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2757
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2813
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2939
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	3045
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	3077
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	3102
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	3219

activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	1285
activemq::wireformat::openwire::marshal::v4::ReplyCommandMarshaller	1317
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1348
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1378
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1423
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1452
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1485
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1514
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1548
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller	1587
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1753
activemq::wireformat::openwire::marshal::v5::ActiveMQBlockWireFormatMarshaller	1783
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	1855
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1961
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	2115
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2168
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	2189
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	2236
activemq::wireformat::openwire::marshal::v5::ActiveMQStackingMessageMarshaller	2255
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	2286
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2317
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2364
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2586
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2615
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2656
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2677
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2694

activemq::wireformat::openwire::marshal::v5::MessagePublishInfoFormat	2749	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp	630
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshal	2805	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp	659
activemq::wireformat::openwire::marshal::v5::PartialCommandFormat	2931	activemq::wireformat::openwire::marshal::v6::ActiveMQTextM	684
activemq::wireformat::openwire::marshal::v5::ProducerIdWireFormat	3053	activemq::wireformat::openwire::marshal::v6::ActiveMQTopicC	712
activemq::wireformat::openwire::marshal::v5::ProducerIdWireFormat	3085	activemq::wireformat::openwire::marshal::v6::BaseCommandM	794
activemq::wireformat::openwire::marshal::v5::ProducerIdWireFormat	3114	activemq::wireformat::openwire::marshal::v6::BrokerIdMarsha	889
activemq::wireformat::openwire::marshal::v5::RemoteInfoWireFormat	3215	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarsh	921
activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshal	3244	activemq::wireformat::openwire::marshal::v6::ConnectionCont	1289
activemq::wireformat::openwire::marshal::v5::ReplaceCommandFormat	3275	activemq::wireformat::openwire::marshal::v6::ConnectionError	1321
activemq::wireformat::openwire::marshal::v5::ResponseWireFormat	3306	activemq::wireformat::openwire::marshal::v6::ConnectionIdMa	1352
activemq::wireformat::openwire::marshal::v5::SessionIdWireFormat	3396	activemq::wireformat::openwire::marshal::v6::ConnectionInfoM	1382
activemq::wireformat::openwire::marshal::v5::SessionInfoWireFormat	3416	activemq::wireformat::openwire::marshal::v6::ConsumerContro	1427
activemq::wireformat::openwire::marshal::v5::ShutdownInfoFormat	3486	activemq::wireformat::openwire::marshal::v6::ConsumerIdMar	1456
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoFormat	3685	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMa	1489
activemq::wireformat::openwire::marshal::v5::TransactionIdWireFormat	3824	activemq::wireformat::openwire::marshal::v6::ControlComman	1518
activemq::wireformat::openwire::marshal::v5::TransactionInfoFormat	3852	activemq::wireformat::openwire::marshal::v6::DataArrayRespo	1552
activemq::wireformat::openwire::marshal::v5::WireFormatInfoFormat	3993	activemq::wireformat::openwire::marshal::v6::DataResponseM	1591
activemq::wireformat::openwire::marshal::v5::XATransactionInfoFormat	4057	activemq::wireformat::openwire::marshal::v6::DestinationInfoM	1749
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormat	230	activemq::wireformat::openwire::marshal::v6::DiscoveryEventM	1763
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshal	266	activemq::wireformat::openwire::marshal::v6::ExceptionRespor	1843
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationFormat	354	activemq::wireformat::openwire::marshal::v6::FlushCommandI	1941
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshal	395	activemq::wireformat::openwire::marshal::v6::IntegerResponse	2095
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormat	422	activemq::wireformat::openwire::marshal::v6::JournalQueueAc	2160
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectWireMessageMarshal	467	activemq::wireformat::openwire::marshal::v6::JournalTopicAck	2201
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormat	512	activemq::wireformat::openwire::marshal::v6::JournalTraceMa	2224
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshal	573	activemq::wireformat::openwire::marshal::v6::JournalTransact	2243
activemq::wireformat::openwire::marshal::v6::ActiveMQTempInfoFormat	601	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoM	2270

- activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 814
- 2305
- activemq::wireformat::openwire::marshal::v6::LooseMarshaller, 814
- 2352
- activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 814
- 2566
- looseMarshalNestedObject
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 815
- 2627
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2893
- 2636
- activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2685
- 2685
- looseMarshalObjectArray
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2719
- 2719
- looseMarshalString
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2765
- 2765
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2801
- 2801
- looseCmmarsh
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- 2923
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3057
- 3057
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3089
- 3089
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3122
- 3122
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3203
- 3203
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3240
- 3240
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3271
- 3271
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3321
- 3321
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3392
- 3392
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3412
- 3412
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3474
- 3474
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3693
- 3693
- activemq::wireformat::openwire::marshal::v6::TransactionMarshaller, 3844
- 3844
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3864
- 3864
- activemq::wireformat::openwire::marshal::v6::WireFormatIdMarshaller, 3997
- 3997
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4037
- 4037
- looseMarshalBrokerError
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 814
- 814
- looseMarshalCachedObject
- 913

activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2761
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	1281
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFactory	1313
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2817
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	1344
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2944
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	1374
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	3061
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	1419
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	3093
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1448
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3110
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1481
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3211
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	1510
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	3228
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	1545
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	3259
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	1608
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3317
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	1745
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3404
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	1779
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3420
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	1864
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3482
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	1957
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3681
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	2112
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3828
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	2180
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3856
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	2209
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	4009
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	2232
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	4049
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	2263
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	222
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	2290
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	270
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	2326
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	350
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	2372
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	391
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2582
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	418
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	2623
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	463
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2652
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	508
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	2689
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	569
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2714
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	597





activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	1949
334	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	2104
375	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	2172
402	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2197
447	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2220
492	
activemq::wireformat::openwire::marshal::v3::ActiveMQStackedMessageMarshaller	2251
553	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	2278
581	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2310
610	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2360
639	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2574
668	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2611
696	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	2644
767	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2681
873	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2699
905	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2753
1273	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2809
1305	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2936
1336	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	3049
1366	
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	3081
1411	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	3118
1440	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	3207
1473	
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	3232
1502	
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	3267
1537	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3312
1600	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	3400
1737	
activemq::wireformat::openwire::marshal::v3::DiscardEventMarshaller	3424
1771	
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	3490
1852	
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v3::NetworkBridgeMarshaller	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	

activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	1477
3677	
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	1506
3836	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1541
3860	
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	1604
4013	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1741
4053	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMarshaller	1775
218	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	1860
258	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1953
342	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	2108
383	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2176
410	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	2205
455	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	2228
500	
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMarshaller	2259
561	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	2282
589	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2322
618	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2368
643	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2578
672	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2619
700	
activemq::wireformat::openwire::marshal::v4::BaseCommandWireFormatMarshaller	2648
774	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2673
877	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2709
909	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2757
1277	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2813
1309	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2940
1340	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	3045
1370	
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	3077
1415	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	3102
1444	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	



activemq::wireformat::openwire::marshal::v5::MessageMarshaller	601	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp
2694		
activemq::wireformat::openwire::marshal::v5::MessagePublishFormat	630	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp
2749		
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	659	activemq::wireformat::openwire::marshal::v6::ActiveMQTemp
2805		
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	684	activemq::wireformat::openwire::marshal::v6::ActiveMQTextM
2932		
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	712	activemq::wireformat::openwire::marshal::v6::ActiveMQTopicC
3053		
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	795	activemq::wireformat::openwire::marshal::v6::BaseCommandM
3085		
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	889	activemq::wireformat::openwire::marshal::v6::BrokerIdMarsh
3114		
activemq::wireformat::openwire::marshal::v5::RemoteInfoMarshaller	921	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarsh
3215		
activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshaller	1289	activemq::wireformat::openwire::marshal::v6::ConnectionCont
3244		
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	1321	activemq::wireformat::openwire::marshal::v6::ConnectionError
3275		
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	1352	activemq::wireformat::openwire::marshal::v6::ConnectionIdMa
3307		
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	1382	activemq::wireformat::openwire::marshal::v6::ConnectionInfoM
3396		
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	1427	activemq::wireformat::openwire::marshal::v6::ConsumerContro
3416		
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	1456	activemq::wireformat::openwire::marshal::v6::ConsumerIdMar
3486		
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	1489	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMa
3685		
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1518	activemq::wireformat::openwire::marshal::v6::ControlComman
3824		
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1553	activemq::wireformat::openwire::marshal::v6::DataArrayRespo
3852		
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	1592	activemq::wireformat::openwire::marshal::v6::DataResponseM
3993		
activemq::wireformat::openwire::marshal::v5::XATransactionMarshaller	1749	activemq::wireformat::openwire::marshal::v6::DestinationInfoM
4057		
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	1763	activemq::wireformat::openwire::marshal::v6::DiscoveryEventM
230		
activemq::wireformat::openwire::marshal::v6::ActiveMQByteWireFormatMarshaller	1844	activemq::wireformat::openwire::marshal::v6::ExceptionRespo
266		
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	1941	activemq::wireformat::openwire::marshal::v6::FlushCommandI
354		
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	2096	activemq::wireformat::openwire::marshal::v6::IntegerResponse
395		
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	2160	activemq::wireformat::openwire::marshal::v6::JournalQueueAc
422		
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectWireMessageMarshaller	2201	activemq::wireformat::openwire::marshal::v6::JournalTopicAck
467		
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	2224	activemq::wireformat::openwire::marshal::v6::JournalTraceMa
512		
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	2243	activemq::wireformat::openwire::marshal::v6::JournalTransact
573		

- activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2270
- activemq::wireformat::openwire::marshal::v6::LastActiveCommandMarshaller, 2306
- activemq::wireformat::openwire::marshal::v6::LostTimedOutLocalMsgObject, 2352
- activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2566
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2627
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2636
- activemq::wireformat::openwire::marshal::v6::MessageIDMarshaller, 2685
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2719
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2765
- activemq::wireformat::openwire::marshal::v6::NewURLBridgeStreamMarshaller, 2801
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2924
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3057
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3089
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3122
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3203
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3240
- activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, 3271
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3322
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3392
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3412
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3474
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3693
- activemq::wireformat::openwire::marshal::v6::TransportIdMarshaller, 3844
- activemq::wireformat::openwire::marshal::v6::TransportInfoMarshaller, 3864
- activemq::wireformat::openwire::marshal::v6::MarkBlockInfoMarshaller, 3997
- activemq::wireformat::openwire::marshal::v6::MarkBlockLoggerMarshaller, 4037
- looseUnmarshalBrokerError, 2483
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 2483

- decaf::io::BufferedInputStream, 937
- decaf::io::ByteArrayInputStream, 1024
- decaf::io::FilterInputStream, 1898
- decaf::io::InputStream, 2046
- decaf::io::PushbackInputStream, 3144
- decaf::io::Reader, 3165
- decaf::util::zip::InflaterInputStream, 2040
- marshal
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3003
  - activemq:wireformat:openwire::OpenWireFormat, 2894
  - activemq:wireformat:openwire::utils::BooleanStream, 858
  - activemq:wireformat:stomp::StompWireFormat, 3645
  - activemq:wireformat::WireFormat, 3978
- marshalledProperties
  - activemq:commands::Message, 2532
- marshalledSize
  - activemq:wireformat:openwire::utils::BooleanStream, 858
- MarshallingSupport
  - activemq:util::MarshallingSupport, 2494
- marshalList
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3003
- marshalMap
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3004
- marshalPrimitive
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3004
- marshalPrimitiveList
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3004
- marshalPrimitiveMap
  - activemq:wireformat:openwire:marshal::PrimitiveTypesMarshaller, 3004
- masterBroker
  - activemq:commands::BrokerInfo, 903
- MATCH
  - inflate.h, 4733
- match\_available
  - internal\_state, 2120
- match\_length
  - internal\_state, 2120
- match\_start
  - internal\_state, 2120
- matches
  - internal\_state, 2120
- Math
  - decaf::lang::Math, 2499
- max
  - decaf::lang::Math, 2501, 2502
- MAX\_BITS
  - deflate.h, 4727
- max\_chain\_length
  - internal\_state, 2120
- max\_code
  - tree\_desc\_s, 3905
- MAX\_DIST
  - deflate.h, 4727
- max\_insert\_length
  - deflate.h, 4727
- max\_lazy\_match
  - internal\_state, 2120
- MAX\_MATCH
  - zutil.h, 4745
- MAX\_MEM\_LEVEL
  - zconf.h, 4738
- MAX\_PREFETCH\_SIZE
  - activemq:core::policies::DefaultPrefetchPolicy, 1676
- MAX\_PRIORITY
  - decaf::lang::Thread, 3773
- MAX\_RADIX
  - decaf::lang::Character, 1106
- MAX\_VALUE
  - decaf::lang::Character, 1106
  - decaf::lang::Double, 1798
  - decaf::lang::Integer, 2090
  - decaf::lang::Long, 2433
  - decaf::lang::Short, 3448
- MAX\_WBITS
  - zconf.h, 4738
- maxValuePerDispatchMessageLimit
  - activemq:commands::ConsumerInfo, 1466
- MEM
  - inflate.h, 4733
- MemoryUsage
  - activemq:util::MemoryUsage, 2513
- MESSAGE
  - activemq:wireformat:stomp::StompCommandConstants, 3631
- Message
  - activemq:commands::Message, 2520
- message
  - activemq:cmsutil::CmsTemplate::ReceiveExecutor, 3176
  - activemq:commands::JournalTrace, 2214
  - activemq:commands::MessageDispatch, 2598
  - decaf::lang::Exception, 1837
- MessageAck
  - activemq:commands::MessageAck, 2560

- messageAck
  - activemq::commands::JournalQueueAck, 2158
- MessageAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2582
  - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2570
  - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2574
  - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2578
  - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2586
  - activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2566
- messageCount
  - activemq::commands::MessageAck, 2564
- MessageDispatch
  - activemq::commands::MessageDispatch, 2595
- MessageDispatchChannel
  - activemq::core::MessageDispatchChannel, 2600
- MessageDispatchMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2623
  - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2607
  - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2611
  - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2619
  - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2615
  - activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2627
- MessageDispatchNotification
  - activemq::commands::MessageDispatchNotification, 2631
- MessageDispatchNotificationMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2652
  - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2640
  - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2644
  - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2648
  - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2656
  - activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2636
- MessageEOFException
  - cms::MessageEOFException, 2661
- MessageFormatException
  - cms::MessageFormatException, 2662
- MessageId
  - activemq::commands::MessageId, 2664
- MessageIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2689
  - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2669
  - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2681
  - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2673
  - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2677
  - activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2685
- MessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2754
- MessageNotReadableException
  - cms::MessageNotReadableException, 2723
- MessageNotWriteableException
  - cms::MessageNotWriteableException, 2724
- MessagePropertyInterceptor
  - activemq::commands::MessagePropertyInterceptor, 2733
- MessagePull
  - activemq::commands::MessagePull, 2740
- MessagePullMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2754
  - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2754



- activemq::wireformat::openwire::marshal::v4::MessagePublicMarshaller
  - 2757
  - decaf::util::Marshaller, 1255
- activemq::wireformat::openwire::marshal::v5::MessagePublicMarshaller
  - 2749
  - decaf::util::Marshaller, 1255
  - MutexHandle, 2787
- activemq::wireformat::openwire::marshal::v6::MessagePublicMarshaller
  - 2765
  - decaf::util::Marshaller, 1255
  - MutexHandle, 2787
- messageSequenceId
  - activemq::commands::JournalTopicAck, 2187
- method
  - internal\_state, 2120
- MethodName
  - activemq::commands::BrokerError::StackTraceElement, 3578
- MICROSECONDS
  - decaf::util::concurrent::TimeUnit, 3815
- MILLISECONDS
  - decaf::util::concurrent::TimeUnit, 3815
- min
  - decaf::lang::Math, 2503, 2504
- MIN\_LOOKAHEAD
  - deflate.h, 4727
- MIN\_MATCH
  - zutil.h, 4745
- MIN\_PRIORITY
  - decaf::lang::Thread, 3773
- MIN\_RADIX
  - decaf::lang::Character, 1106
- MIN\_VALUE
  - decaf::lang::Byte, 968
  - decaf::lang::Character, 1107
  - decaf::lang::Double, 1798
  - decaf::lang::Float, 1914
  - decaf::lang::Integer, 2090
  - decaf::lang::Long, 2433
  - decaf::lang::Short, 3448
- MINUTES
  - decaf::util::concurrent::TimeUnit, 3815
- MockTransport
  - activemq::transport::mock::MockTransport, 2770
- mode
  - gz\_state, 1977
  - inflate\_state, 2025
- modifiedUtf8ToAscii
  - activemq::util::MarshallingSupport, 2494
- msg
  - gz\_state, 1977
  - z\_stream\_s, 4062
- Mutex
  - decaf::util::concurrent::Mutex, 2783
- mutex
  - decaf::util::AbstractCollection, 191
- NAME
  - inflate.h, 4732
- name
  - gz\_header\_s, 1975
- name\_max
  - gz\_header\_s, 1975
- NAME\_STATE
  - deflate.h, 4727
- nameUUIDFromBytes
  - decaf::util::UUID, 3972
- NaN
  - decaf::lang::Double, 1798
  - decaf::lang::Float, 1914
- NANOSECONDS
  - decaf::util::concurrent::TimeUnit, 3815
- nanoTime
  - decaf::lang::System, 3732
- narrow
  - activemq::transport::failover::FailoverTransport, 1879
  - activemq::transport::IOTransport, 2148
  - activemq::transport::mock::MockTransport, 2773
  - activemq::transport::Transport, 3885
  - activemq::transport::TransportFilter, 3895
- ncode
  - inflate\_state, 2025
- ndist
  - inflate\_state, 2025
- needsDictionary
  - decaf::util::zip::Inflater, 2031
- needsInput
  - decaf::util::zip::Deflater, 1711
  - decaf::util::zip::Inflater, 2032
- NEGATIVE\_INFINITY
  - decaf::lang::Double, 1798
  - decaf::lang::Float, 1914
- Network
  - decaf::internal::net::Network, 2791
- NetworkBridgeFilter
  - activemq::commands::NetworkBridgeFilter, 2794
- NetworkBridgeFilterMarshaller
  - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilter, 2817
  - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilter, 2797

- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshal, 2845
- 2809
- nextInt
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshal, 2813
- nextLong
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshal, 2805
- nextMessage
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshal, 2801
- 489
- networkBrokerId
- activemq::commands::NetworkBridgeFilter, 2795
- networkConnection
- activemq::commands::BrokerInfo, 903
- networkConsumerPath
- activemq::commands::ConsumerInfo, 1466
- networkProperties
- activemq::commands::BrokerInfo, 903
- networkSubscription
- activemq::commands::ConsumerInfo, 1466
- networkTTL
- activemq::commands::NetworkBridgeFilter, 2795
- NEW
- decaf::lang::Thread, 3768
- newCondition
- decaf::util::concurrent::locks::Lock, 2379
- decaf::util::concurrent::locks::ReentrantLock, 3185
- newThread
- decaf::util::concurrent::ThreadFactory, 3774
- next
- activemq::transport::TransportFilter, 3898
- decaf::security::SecureRandom, 3333
- decaf::util::Iterator, 2154
- decaf::util::Random, 3156
- gz\_state, 1977
- inflate\_state, 2025
- next\_in
- z\_stream\_s, 4062
- next\_out
- z\_stream\_s, 4062
- nextBoolean
- decaf::util::Random, 3157
- nextBytes
- decaf::security::SecureRandom, 3333, 3334
- decaf::util::Random, 3157
- nextDouble
- decaf::util::Random, 3157
- nextFloat
- decaf::util::Random, 3158
- nextGaussian
- decaf::util::Random, 3158
- nextIndex
- cms::MessageEnumeration, 2659
- nextToken
- decaf::util::StringTokenizer, 3669
- nice\_match
- internal\_state, 2120
- nlen
- inflate\_state, 2025
- NO\_COMPRESSION
- decaf::util::zip::Deflater, 1715
- NO\_MAXIMUM\_REDELIVERIES
- activemq::core::RedeliveryPolicy, 3181
- node
- decaf::util::UUID, 3972
- noLocal
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3176
- activemq::commands::ConsumerInfo, 1466
- NON\_PERSISTENT
- cms::DeliveryMode, 1721
- noRangeAcks
- activemq::commands::ConsumerInfo, 1466
- NORM\_PRIORITY
- decaf::lang::Thread, 3773
- normalize
- decaf::net::URI, 3928
- NoRouteToHostException
- decaf::net::NoRouteToHostException, 2820, 2821
- NoSuchAlgorithmException
- decaf::security::NoSuchAlgorithmException, 2823, 2824
- NoSuchElementException
- decaf::lang::exceptions::NoSuchElementException, 2826, 2827
- NoSuchProviderException
- decaf::security::NoSuchProviderException, 2829, 2830
- notify
- activemq::core::MessageDispatchChannel, 2602
- decaf::internal::util::concurrent::ConditionImpl, 1257
- decaf::internal::util::concurrent::SynchronizableImpl, 3712
- decaf::io::InputStream, 2047
- decaf::io::OutputStream, 2910

- decaf::util::AbstractCollection, 186
- decaf::util::concurrent::ConcurrentStlMap, 1241
- decaf::util::concurrent::Mutex, 2783
- decaf::util::concurrent::Synchronizable, 3702
- decaf::util::StlMap, 3609
- decaf::util::StlQueue, 3619
- notifyAll
  - activemq::core::MessageDispatchChannel, 2602
  - decaf::internal::util::concurrent::ConditionImpl, 1258
  - decaf::internal::util::concurrent::SynchronizableImpl, 3712
  - decaf::io::InputStream, 2047
  - decaf::io::OutputStream, 2910
  - decaf::util::AbstractCollection, 186
  - decaf::util::concurrent::ConcurrentStlMap, 1241
  - decaf::util::concurrent::Mutex, 2783
  - decaf::util::concurrent::Synchronizable, 3703
  - decaf::util::StlMap, 3610
  - decaf::util::StlQueue, 3619
- Null
  - decaf::util::logging, 176
- NULL\_TYPE
  - activemq::util::PrimitiveValueNode, 3015
  - activemq::wireformat::openwire::OpenWireFormat, 2898
- NullPointerException
  - decaf::lang::exceptions::NullPointerException, 2832, 2833
- NUM\_OPTIONS
  - activemq::core::ActiveMQConstants, 308
- NUM\_PARAMS
  - activemq::core::ActiveMQConstants, 309
- NumberFormatException
  - decaf::lang::exceptions::NumberFormatException, 2838, 2839
- numberOfLeadingZeros
  - decaf::lang::Integer, 2083
  - decaf::lang::Long, 2426
- numberOfTrailingZeros
  - decaf::lang::Integer, 2083
  - decaf::lang::Long, 2426
- numWaiting
  - decaf::util::concurrent::ConditionHandle, 1255
- numWake
  - decaf::util::concurrent::ConditionHandle, 1255
- objectId
  - activemq::commands::RemoveInfo, 3197
- OF
  - deflate.h, 4727
  - gzguts.h, 4729
  - inffast.h, 4730
  - infrees.h, 4734
  - zconf.h, 4738
  - zlib.h, 4742
  - zutil.h, 4745
- OFF
  - decaf::util::logging::Level, 2336
- Off
  - decaf::util::logging, 176
- offer
  - decaf::util::concurrent::BlockingQueue, 847
  - decaf::util::concurrent::SynchronousQueue, 3721
  - decaf::util::PriorityQueue, 3032
  - decaf::util::Queue, 3150
- offset
  - decaf::internal::nio::CharArrayBuffer, 1118
  - inflate\_state, 2025
- onCommand
  - activemq::core::ActiveMQConnection, 285
  - activemq::transport::correlator::ResponseCorrelator, 3292
  - activemq::transport::DefaultTransportListener, 1704
  - activemq::transport::failover::FailoverTransportListener, 1889
  - activemq::transport::inactivity::InactivityMonitor, 2005
  - activemq::transport::logging::LoggingTransport, 2404
  - activemq::transport::mock::InternalCommandListener, 2122
  - activemq::transport::TransportFilter, 3895
  - activemq::transport::TransportListener, 3900
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2902
- oneway
  - activemq::core::ActiveMQConnection, 285
  - activemq::core::ActiveMQSession, 529
  - activemq::transport::correlator::ResponseCorrelator, 3292
  - activemq::transport::failover::FailoverTransport, 1880
  - activemq::transport::inactivity::InactivityMonitor, 2006
  - activemq::transport::IOTransport, 2148
  - activemq::transport::logging::LoggingTransport, 2404

- activemq::transport::mock::MockTransport, 2773
- activemq::transport::Transport, 3886
- activemq::transport::TransportFilter, 3895
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2902
- onException
  - activemq::core::ActiveMQConnection, 285
  - activemq::transport::DefaultTransportListener, 1704
  - activemq::transport::failover::BackupTransport, 753
  - activemq::transport::failover::FailoverTransportListener, 1889
  - activemq::transport::inactivity::InactivityMonitor, 2006
  - activemq::transport::TransportFilter, 3896
  - activemq::transport::TransportListener, 3901
  - cms::ExceptionListener, 1838
- onMessage
  - cms::MessageListener, 2692
- onProducerAck
  - activemq::core::ActiveMQProducer, 474
- onPropertiesReset
  - decaf::util::logging::PropertiesChangeListener, 3135
- onPropertyChanged
  - decaf::util::logging::PropertiesChangeListener, 3135
- onResponse
  - activemq::state::Tracked, 3818
- onSend
  - activemq::commands::ActiveMQBytesMessage, 238
  - activemq::commands::ActiveMQMessageTemplate, 435
  - activemq::commands::ActiveMQStreamMessage, 541
  - activemq::commands::Message, 2527
- onTaskComplete
  - decaf::util::concurrent::TaskListener, 3735
- onTaskCompleted
  - decaf::util::concurrent::PooledThreadListener, 2971
  - decaf::util::concurrent::ThreadPool, 3780
- onTaskException
  - decaf::util::concurrent::PooledThreadListener, 2972
  - decaf::util::concurrent::TaskListener, 3735
  - decaf::util::concurrent::ThreadPool, 3780
- onTaskStarted
  - decaf::util::concurrent::PooledThreadListener, 2972
- decaf::util::concurrent::ThreadPool, 3781
- onTransportException
  - activemq::transport::correlator::ResponseCorrelator, 3293
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2903
- op
  - code, 1183
- opaque
  - z\_stream\_s, 4062
- OPEN\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- OpenSSLContextSpi
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2843
- OpenSSLServerSocket
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2849
- OpenSSLServerSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2855
- OpenSSLSocket
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2844
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2862
- OpenSSLSocketException
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2872, 2873
- OpenSSLSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2844
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2877
- OpenSSLSocketInputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2883
- OpenSSLSocketOutputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2886
- OpenWireFormat
  - activemq::wireformat::openwire::OpenWireFormat, 2890
- OpenWireFormatFactory
  - activemq::wireformat::openwire::OpenWireFormatFactory, 2899
- OpenWireFormatNegotiator
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2901
- OpenWireResponseBuilder
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2905
- operationType

- activemq::commands::DestinationInfo, 1731
- operator<
  - activemq::commands::BrokerId, 871
  - activemq::commands::ConnectionId, 1329
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::LocalTransactionId, 2349
  - activemq::commands::MessageId, 2666
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionId, 3382
  - activemq::commands::TransactionId, 3821
  - activemq::commands::XATransactionId, 4034
  - decaf::lang::Boolean, 852
  - decaf::lang::Byte, 964
  - decaf::lang::Character, 1105
  - decaf::lang::Comparable, 1215
  - decaf::lang::Double, 1794, 1795
  - decaf::lang::Float, 1910, 1911
  - decaf::lang::Integer, 2084
  - decaf::lang::Long, 2427
  - decaf::lang::Short, 3444
  - decaf::net::URI, 3928
  - decaf::nio::ByteBuffer, 1046
  - decaf::nio::CharBuffer, 1128
  - decaf::nio::DoubleBuffer, 1815
  - decaf::nio::FloatBuffer, 1931
  - decaf::nio::IntBuffer, 2072
  - decaf::nio::LongBuffer, 2450
  - decaf::nio::ShortBuffer, 3465
  - decaf::util::concurrent::TimeUnit, 3810
  - decaf::util::Date, 1667
  - decaf::util::logging::Level, 2334
  - decaf::util::UUID, 3973
- operator\*
- decaf::lang::Pointer, 2950
- operator()
  - decaf::lang::ArrayPointerComparator, 737
  - decaf::lang::PointerComparator, 2954
  - decaf::util::Comparator, 1218
  - decaf::util::comparators::Less, 2329
  - std::less< decaf::lang::ArrayPointer< T > >, 2330
  - std::less< decaf::lang::Pointer< T > >, 2331
- operator->
  - decaf::lang::Pointer, 2951
- operator=
  - activemq::cmsutil::CachedConsumer, 1072
  - activemq::cmsutil::CachedProducer, 1078
  - activemq::cmsutil::CmsAccessor, 1156
  - activemq::cmsutil::CmsDestinationAccessor, 1159
  - activemq::cmsutil::CmsTemplate, 1176
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3066
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
  - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3278
  - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3279
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3350
  - activemq::cmsutil::DynamicDestinationResolver, 1822
  - activemq::cmsutil::PooledSession, 2965
  - activemq::cmsutil::ResourceLifecycleManager, 3284
  - activemq::cmsutil::SessionPool, 3436
  - activemq::commands::BrokerId, 871
  - activemq::commands::ConnectionId, 1329
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::LocalTransactionId, 2349
  - activemq::commands::MessageId, 2666
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionId, 3382
  - activemq::commands::TransactionId, 3821
  - activemq::commands::XATransactionId, 4034
  - activemq::library::ActiveMQCPP, 321
  - activemq::util::PrimitiveValueNode, 3022
  - decaf::lang::ArrayPointer, 734
  - decaf::lang::Exception, 1836
  - decaf::lang::Pointer, 2951
  - decaf::util::AbstractCollection, 187
  - decaf::util::Date, 1667
  - decaf::util::logging::LogManager, 2410
  - decaf::util::PriorityQueue, 3032, 3033
  - decaf::util::Properties, 3131
- operator==
  - activemq::commands::BrokerId, 871
  - activemq::commands::ConnectionId, 1329
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::LocalTransactionId, 2349
  - activemq::commands::MessageId, 2666
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionId, 3382
  - activemq::commands::TransactionId, 3821
  - activemq::commands::XATransactionId, 4034
  - activemq::util::PrimitiveValueNode, 3022
  - decaf::lang, 158
  - decaf::lang::ArrayPointer, 734, 736
  - decaf::lang::Boolean, 853

- decaf::lang::Byte, 965
- decaf::lang::Character, 1105
- decaf::lang::Comparable, 1215
- decaf::lang::Double, 1795
- decaf::lang::Float, 1911
- decaf::lang::Integer, 2084
- decaf::lang::Long, 2427
- decaf::lang::Pointer, 2951, 2953
- decaf::lang::Short, 3445
- decaf::net::URI, 3929
- decaf::nio::ByteBuffer, 1046
- decaf::nio::CharBuffer, 1128
- decaf::nio::DoubleBuffer, 1816
- decaf::nio::FloatBuffer, 1931
- decaf::nio::IntBuffer, 2072
- decaf::nio::LongBuffer, 2450
- decaf::nio::ShortBuffer, 3465
- decaf::util::concurrent::TimeUnit, 3810
- decaf::util::Date, 1668
- decaf::util::logging::Level, 2334
- decaf::util::UUID, 3973
- operator[]
  - activemq::wireformat::openwire::utils::HexTable, 1984
  - decaf::internal::util::ByteArrayAdapter, 983
  - decaf::lang::ArrayPointer, 735
- opt\_len
  - internal\_state, 2120
- optimizedAcknowledge
  - activemq::commands::ConsumerInfo, 1466
- options
  - activemq::commands::ActiveMQDestination, 332
- ordered
  - activemq::commands::ActiveMQDestination, 332
- orderedTarget
  - activemq::commands::ActiveMQDestination, 332
- originalDestination
  - activemq::commands::Message, 2532
- originalTransactionId
  - activemq::commands::Message, 2532
- OS
  - inflate.h, 4732
- os
  - gz\_header\_s, 1975
- OS\_CODE
  - zutil.h, 4745
- out
  - decaf::io::FileDescriptor, 1892
  - gz\_state, 1977
- OutputStream
  - decaf::io::OutputStream, 2909
- outputStream
  - decaf::io::FilterOutputStream, 1903
- OutputStreamWriter
  - decaf::io::OutputStreamWriter, 2915
- own
  - decaf::io::FilterInputStream, 1899
  - decaf::io::FilterOutputStream, 1903
- ownDeflater
  - decaf::util::zip::DeflaterOutputStream, 1719
- ownInflater
  - decaf::util::zip::InflaterInputStream, 2041
- PARAM\_CLIENTID
  - activemq::core::ActiveMQConstants, 309
- PARAM\_PASSWORD
  - activemq::core::ActiveMQConstants, 309
- PARAM\_USERNAME
  - activemq::core::ActiveMQConstants, 309
- parent
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3066
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3176
- park
  - decaf::util::concurrent::locks::LockSupport, 2384
- parkNanos
  - decaf::util::concurrent::locks::LockSupport, 2384
- parkUntil
  - decaf::util::concurrent::locks::LockSupport, 2385
- parse
  - decaf::internal::util::HexStringParser, 1982
  - decaf::util::logging::Level, 2334
- parseAuthority
  - decaf::internal::net::URIHelper, 3938
- parseBoolean
  - decaf::lang::Boolean, 853
- parseByte
  - decaf::lang::Byte, 965
- parseComposite
  - activemq::util::URISupport, 3945
- parseDouble
  - decaf::internal::util::HexStringParser, 1982
  - decaf::lang::Double, 1795
- parseFloat
  - decaf::internal::util::HexStringParser, 1983
  - decaf::lang::Float, 1911
- parseInt
  - decaf::lang::Integer, 2085
- parseLong

- decaf::lang::Long, 2428
- parseQuery
  - activemq::util::URISupport, 3946
- parseServerAuthority
  - decaf::net::URI, 3929
- parseShort
  - decaf::lang::Short, 3445
- parseURI
  - decaf::internal::net::URIHelper, 3939
- parseURL
  - activemq::util::URISupport, 3946
- PartialCommand
  - activemq::commands::PartialCommand, 2919
- PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2943
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2927
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2935
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2939
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2931
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- password
  - activemq::commands::ConnectionInfo, 1360
- path
  - gz\_state, 1977
- peek
  - activemq::core::MessageDispatchChannel, 2603
  - decaf::internal::util::TimerTaskHeap, 3805
  - decaf::util::concurrent::SynchronousQueue, 3722
  - decaf::util::PriorityQueue, 3033
  - decaf::util::Queue, 3150
- peerBrokerInfos
  - activemq::commands::BrokerInfo, 903
- pending
  - internal\_state, 2120
- pending\_buf
  - internal\_state, 2120
- pending\_buf\_size
  - internal\_state, 2120
- pending\_out
  - internal\_state, 2120
- PERSISTENT
  - cms::DeliveryMode, 1721
- persistent
  - activemq::commands::Message, 2532
- physicalName
  - activemq::commands::ActiveMQDestination, 332
- PI
  - decaf::lang::Math, 2511
- Pointer
  - decaf::lang::Pointer, 2948, 2949
- PointerType
  - decaf::lang::ArrayPointer, 732
  - decaf::lang::Pointer, 2948
- poll
  - decaf::util::concurrent::BlockingQueue, 847
  - decaf::util::concurrent::SynchronousQueue, 3722
  - decaf::util::PriorityQueue, 3033
- PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2943
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2927
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2935
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2939
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2931
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- PooledSession
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2943
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2927
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2935
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2939
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2931
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- PooledThread
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2943
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2927
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2935
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2939
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2931
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2923
- pop
  - decaf::util::concurrent::BlockingQueue, 847
  - decaf::util::concurrent::SynchronousQueue, 3722
  - decaf::util::PriorityQueue, 3033
- port
  - decaf::lang::Math, 2511
- PortUnreachableException
  - activemq::commands::ActiveMQDestination, 332
- Pos
  - deflate.h, 4727
- pos
  - gz\_state, 1977
- Posf
  - deflate.h, 4727
- position
  - decaf::nio::Buffer, 932
- POSITIVE\_INFINITY
  - decaf::lang::Double, 1798
  - decaf::lang::Float, 1914
- pow
  - decaf::lang::Math, 2505
- prefetch
  - activemq::commands::ConsumerControl, 1405
- PrefetchPolicy
  - activemq::core::PrefetchPolicy, 2977
- prefetchSize
  - activemq::commands::ConsumerInfo, 1466
- PRESET\_DICT
  - zutil.h, 4745
- prev
  - internal\_state, 2120
- prev\_length
  - internal\_state, 2120
- prev\_match
  - internal\_state, 2120

- internal\_state, 2120
- previous
  - decaf::util::ListIterator, 2345
- previousIndex
  - decaf::util::ListIterator, 2346
- PrimitiveList
  - activemq::util::PrimitiveList, 2982
- PrimitiveMap
  - activemq::util::PrimitiveMap, 2993
- PrimitiveType
  - activemq::util::PrimitiveValueNode, 3015
- PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3003
- PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 3010
- PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 3016–3018
- printStackTrace
  - cms::CMSException, 1162
  - decaf::lang::Exception, 1836
  - decaf::lang::Throwable, 3786
- priority
  - activemq::commands::ConsumerInfo, 1466
  - activemq::commands::Message, 2532
- PriorityQueue
  - decaf::util::PriorityQueue, 3030, 3031
- PriorityQueueIterator
  - decaf::util::PriorityQueue, 3035
- processBeginTransaction
  - activemq::state::CommandVisitor, 1202
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1394
- processBrokerError
  - activemq::state::CommandVisitor, 1202
  - activemq::state::CommandVisitorAdapter, 1211
- processBrokerInfo
  - activemq::state::CommandVisitor, 1202
  - activemq::state::CommandVisitorAdapter, 1211
- processCommitTransactionOnePhase
  - activemq::state::CommandVisitor, 1202
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1394
- processCommitTransactionTwoPhase
  - activemq::state::CommandVisitor, 1202
- activemq::state::CommandVisitorAdapter, 1211
- activemq::state::ConnectionStateTracker, 1394
- processConnectionControl
  - activemq::state::CommandVisitor, 1202
  - activemq::state::CommandVisitorAdapter, 1211
- processConnectionError
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
- processConnectionInfo
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processConsumerControl
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
- processConsumerInfo
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processControlCommand
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
- processDestinationInfo
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processEndTransaction
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processFlushCommand
  - activemq::state::CommandVisitor, 1203
  - activemq::state::CommandVisitorAdapter, 1211
- processForgetTransaction
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
- processKeepAliveInfo
  - activemq::state::CommandVisitor, 1204



- activemq::state::CommandVisitorAdapter, 1211
- processMessage
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processMessageAck
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processMessageDispatch
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
- processMessageDispatchNotification
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
- processMessagePull
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
- processPrepareTransaction
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1395
- processProducerAck
  - activemq::state::CommandVisitor, 1204
  - activemq::state::CommandVisitorAdapter, 1211
- processProducerInfo
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1396
- processRecoverTransactions
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1211
- processRemoveConnection
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1396
- processRemoveConsumer
  - activemq::state::CommandVisitor, 1205
- activemq::state::CommandVisitorAdapter, 1211
- activemq::state::ConnectionStateTracker, 1396
- processRemoveDestination
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1211
  - activemq::state::ConnectionStateTracker, 1396
- processRemoveInfo
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1211
- processRemoveProducer
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1212
  - activemq::state::ConnectionStateTracker, 1396
- processRemoveSession
  - activemq::state::CommandVisitor, 1205
  - activemq::state::CommandVisitorAdapter, 1212
  - activemq::state::ConnectionStateTracker, 1396
- processRemoveSubscriptionInfo
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
- processReplayCommand
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
- processResponse
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
- processRollbackTransaction
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
  - activemq::state::ConnectionStateTracker, 1396
- processSessionInfo
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
  - activemq::state::ConnectionStateTracker, 1397
- processShutdownInfo
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212

- processTransactionInfo
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1212
- processWireFormat
  - activemq::state::CommandVisitor, 1206
  - activemq::state::CommandVisitorAdapter, 1213
- PRODUCER\_ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination, 332
- ProducerAck
  - activemq::commands::ProducerAck, 3037
- ProducerAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3061
  - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 3041
  - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 3049
  - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 3045
  - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3053
  - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3057
- ProducerExecutor
  - activemq::cmsutil::CmsTemplate, 1182
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3065
- ProducerId
  - activemq::commands::ProducerId, 3068
- producerId
  - activemq::commands::Message, 2532
  - activemq::commands::MessageId, 2667
  - activemq::commands::ProducerAck, 3039
  - activemq::commands::ProducerInfo, 3100
- ProducerIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3093
  - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3073
  - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3081
  - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3077
  - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3085
  - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3089
- ProducerInfo
  - activemq::commands::ProducerInfo, 3097
- ProducerInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3110
  - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3106
  - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3118
  - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3102
  - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3114
  - activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3122
- producerSequenceId
  - activemq::commands::MessageId, 2667
- ProducerState
  - activemq::state::ProducerState, 3125
- Properties
  - decaf::util::Properties, 3128
- propertyExists
  - activemq::commands::ActiveMQMessageTemplate, 436
  - cms::Message, 2549
- propertyNames
  - decaf::util::Properties, 3132
- ProtocolException
  - decaf::net::ProtocolException, 3137, 3138
- providerGenerateSeed
  - decaf::internal::security::SecureRandomImpl, 3337
  - decaf::security::SecureRandomSpi, 3339
- providerGetDefaultSSLParameters
  - decaf::net::ssl::SSLContextSpi, 3548
- providerGetServerSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2843
  - decaf::net::ssl::SSLContextSpi, 3549
- providerGetSupportedSSLParameters
  - decaf::net::ssl::SSLContextSpi, 3549
- providerInit
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2844
- providerNextBytes
  - decaf::internal::security::SecureRandomImpl, 3337, 3338
  - decaf::security::SecureRandomSpi, 3340
- providerSetSeed
  - decaf::internal::security::SecureRandomImpl, 3338
  - decaf::security::SecureRandomSpi, 3340

- publish
  - decaf::util::logging::ConsoleHandler, 1400
  - decaf::util::logging::Handler, 1980
  - decaf::util::logging::StreamHandler, 3651
- purge
  - decaf::util::Timer, 3792
- push
  - decaf::util::StlQueue, 3619
- PushbackInputStream
  - decaf::io::PushbackInputStream, 3143
- put
  - decaf::internal::nio::ByteBuffer, 1013
  - decaf::internal::nio::CharArrayBuffer, 1116
  - decaf::internal::nio::DoubleArrayBuffer, 1806, 1807
  - decaf::internal::nio::FloatArrayBuffer, 1922, 1923
  - decaf::internal::nio::IntArrayBuffer, 2063, 2064
  - decaf::internal::nio::LongArrayBuffer, 2441, 2442
  - decaf::internal::nio::ShortArrayBuffer, 3456, 3457
  - decaf::internal::util::ByteArrayAdapter, 984
  - decaf::nio::ByteBuffer, 1046–1048
  - decaf::nio::CharBuffer, 1128–1131
  - decaf::nio::DoubleBuffer, 1816, 1817
  - decaf::nio::FloatBuffer, 1931–1933
  - decaf::nio::IntBuffer, 2072–2074
  - decaf::nio::LongBuffer, 2450–2452
  - decaf::nio::ShortBuffer, 3465–3467
  - decaf::util::concurrent::BlockingQueue, 848
  - decaf::util::concurrent::ConcurrentStlMap, 1242
  - decaf::util::concurrent::SynchronousQueue, 3723
  - decaf::util::Map, 2467
  - decaf::util::StlMap, 3610
- put\_byte
  - deflate.h, 4727
- putAll
  - decaf::util::concurrent::ConcurrentStlMap, 1242
  - decaf::util::Map, 2467
  - decaf::util::StlMap, 3610, 3611
- putChar
  - decaf::internal::nio::ByteBuffer, 1013, 1014
  - decaf::internal::util::ByteArrayAdapter, 984
  - decaf::nio::ByteBuffer, 1049
- putDouble
  - decaf::internal::nio::ByteBuffer, 1014, 1015
  - decaf::internal::util::ByteArrayAdapter, 984
  - decaf::nio::ByteBuffer, 1049, 1050
- putDoubleAt
  - decaf::internal::util::ByteArrayAdapter, 985
- putFloat
  - decaf::internal::nio::ByteBuffer, 1015, 1016
  - decaf::internal::util::ByteArrayAdapter, 985
  - decaf::nio::ByteBuffer, 1050, 1051
- putFloatAt
  - decaf::internal::util::ByteArrayAdapter, 986
- putIfAbsent
  - decaf::util::concurrent::ConcurrentMap, 1229
  - decaf::util::concurrent::ConcurrentStlMap, 1243
- putInt
  - decaf::internal::nio::ByteBuffer, 1016
  - decaf::internal::util::ByteArrayAdapter, 986
  - decaf::nio::ByteBuffer, 1051
- putIntAt
  - decaf::internal::util::ByteArrayAdapter, 986
- putLong
  - decaf::internal::nio::ByteBuffer, 1017
  - decaf::internal::util::ByteArrayAdapter, 987
  - decaf::nio::ByteBuffer, 1052
- putLongAt
  - decaf::internal::util::ByteArrayAdapter, 987
- putShort
  - decaf::internal::nio::ByteBuffer, 1018
  - decaf::internal::util::ByteArrayAdapter, 988
  - decaf::nio::ByteBuffer, 1053
- putShortAt
  - decaf::internal::util::ByteArrayAdapter, 988
- QUEUE
  - cms::Destination, 1723
- QUEUE\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- QUEUE\_QUALIFIED\_PREFIX

- activemq::commands::ActiveMQDestination, 332
- queueTask
  - decaf::util::concurrent::ThreadPool, 3781
- quoteIllegal
  - decaf::internal::net::URLEncoderDecoder, 3933
- Random
  - decaf::util::Random, 3156
- random
  - decaf::lang::Math, 2505
- randomUUID
  - decaf::util::UUID, 3973
- raw
  - gz\_state, 1977
- reached
  - decaf::net::InetAddress, 2022
- read
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2866
  - decaf::internal::net::tcp::TcpSocket, 3744
  - decaf::internal::util::ByteArrayAdapter, 988
  - decaf::io::InputStream, 2047–2049
  - decaf::io::Reader, 3165–3167
  - decaf::lang::Readable, 3160
  - decaf::nio::CharBuffer, 1131
- readAsciiString
  - activemq::wireformat::openwire::marshal::BaseDataStreamAdapter, 819
- readBoolean
  - activemq::commands::ActiveMQBytesMessage, 238
  - activemq::commands::ActiveMQStreamMessage, 541
  - activemq::wireformat::openwire::utils::BooleanStream, 858
  - cms::BytesMessage, 1060
  - cms::StreamMessage, 3655
  - decaf::io::DataInput, 1559
  - decaf::io::DataInputStream, 1568
- readByte
  - activemq::commands::ActiveMQBytesMessage, 238
  - activemq::commands::ActiveMQStreamMessage, 542
  - cms::BytesMessage, 1060
  - cms::StreamMessage, 3655
  - decaf::io::DataInput, 1560
  - decaf::io::DataInputStream, 1568
- readBytes
  - activemq::commands::ActiveMQBytesMessage, 239
- activemq::commands::ActiveMQStreamMessage, 542, 543
- cms::BytesMessage, 1061
- cms::StreamMessage, 3655, 3656
- readChar
  - activemq::commands::ActiveMQBytesMessage, 240
  - activemq::commands::ActiveMQStreamMessage, 543
  - cms::BytesMessage, 1062
  - cms::StreamMessage, 3657
  - decaf::io::DataInput, 1560
  - decaf::io::DataInputStream, 1568
- ReadChecker
  - activemq::transport::inactivity::InactivityMonitor, 2007
  - activemq::transport::inactivity::ReadChecker, 3162
- readConfiguration
  - decaf::util::logging::LogManager, 2410, 2411
- readDouble
  - activemq::commands::ActiveMQBytesMessage, 240
  - activemq::commands::ActiveMQStreamMessage, 544
  - cms::BytesMessage, 1062
  - cms::StreamMessage, 3657
  - decaf::io::DataInput, 1560
  - decaf::io::DataInputStream, 1568
- Reader
  - decaf::io::Reader, 3164
- readFloat
  - activemq::commands::ActiveMQBytesMessage, 241
  - activemq::commands::ActiveMQStreamMessage, 544
  - cms::BytesMessage, 1062
  - cms::StreamMessage, 3657
  - decaf::io::DataInput, 1560
  - decaf::io::DataInputStream, 1569
- readFully
  - decaf::io::DataInput, 1561
  - decaf::io::DataInputStream, 1569, 1570
- readInt
  - activemq::commands::ActiveMQBytesMessage, 241
  - activemq::commands::ActiveMQStreamMessage, 544
  - cms::BytesMessage, 1063
  - cms::StreamMessage, 3658
  - decaf::io::DataInput, 1562
  - decaf::io::DataInputStream, 1570
- readLine

- decaf::io::DataInput, 1562
- decaf::io::DataInputStream, 1570
- readLock
  - decaf::util::concurrent::locks::ReadWriteLock, 3173
- readLong
  - activemq::commands::ActiveMQBytesMessage, 241
  - activemq::commands::ActiveMQStreamMessage, 545
  - cms::BytesMessage, 1063
  - cms::StreamMessage, 3658
  - decaf::io::DataInput, 1563
  - decaf::io::DataInputStream, 1571
- readOnly
  - decaf::internal::nio::CharArrayBuffer, 1118
- readonly
  - decaf::io::FileDescriptor, 1892
- ReadOnlyBufferException
  - decaf::nio::ReadOnlyBufferException, 3169, 3170
- readShort
  - activemq::commands::ActiveMQBytesMessage, 242
  - activemq::commands::ActiveMQStreamMessage, 545
  - cms::BytesMessage, 1063
  - cms::StreamMessage, 3659
  - decaf::io::DataInput, 1563
  - decaf::io::DataInputStream, 1571
- readString
  - activemq::commands::ActiveMQBytesMessage, 242
  - activemq::commands::ActiveMQStreamMessage, 546
  - cms::BytesMessage, 1064
  - cms::StreamMessage, 3659
  - decaf::io::DataInput, 1563
  - decaf::io::DataInputStream, 1572
- readString16
  - activemq::util::MarshallingSupport, 2494
- readString32
  - activemq::util::MarshallingSupport, 2495
- readUnsignedByte
  - decaf::io::DataInput, 1564
  - decaf::io::DataInputStream, 1572
- readUnsignedShort
  - activemq::commands::ActiveMQBytesMessage, 242
  - activemq::commands::ActiveMQStreamMessage, 546
  - cms::BytesMessage, 1064
  - cms::StreamMessage, 3659
  - decaf::io::DataInput, 1564
- decaf::io::DataInputStream, 1572
- readUTF
  - activemq::commands::ActiveMQBytesMessage, 243
  - cms::BytesMessage, 1065
  - decaf::io::DataInput, 1564
  - decaf::io::DataInputStream, 1572
- ready
  - decaf::io::InputStreamReader, 2054
  - decaf::io::Reader, 3167
- rebalanceConnection
  - activemq::commands::ConnectionControl, 1271
- RECEIPT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- receive
  - activemq::cmsutil::CachedConsumer, 1073
  - activemq::cmsutil::CmsTemplate, 1176, 1177
  - activemq::core::ActiveMQConsumer, 316, 317
  - cms::MessageConsumer, 2590, 2591
  - RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT
    - activemq::cmsutil::CmsTemplate, 1182
  - RECEIVE\_TIMEOUT\_NO\_WAIT
    - activemq::cmsutil::CmsTemplate, 1182
  - ReceiveExecutor
    - activemq::cmsutil::CmsTemplate, 1182
    - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3175
  - receiveNoWait
    - activemq::cmsutil::CachedConsumer, 1073
    - activemq::core::ActiveMQConsumer, 317
    - cms::MessageConsumer, 2591
- receiveSelected
  - activemq::cmsutil::CmsTemplate, 1177, 1178
- recievedByDFBridge
  - activemq::commands::Message, 2532
- reconnect
  - activemq::transport::failover::FailoverTransport, 1880
  - activemq::transport::IOTransport, 2148
  - activemq::transport::mock::MockTransport, 2774
  - activemq::transport::Transport, 3886
  - activemq::transport::TransportFilter, 3896
- reconnectTo
  - activemq::commands::ConnectionControl, 1271
- recover
  - activemq::cmsutil::PooledSession, 2966
  - activemq::core::ActiveMQSession, 529

- cms::Session, 3375
- redeliveryCounter
  - activemq::commands::Message, 2532
  - activemq::commands::MessageDispatch, 2598
- RedeliveryPolicy
  - activemq::core::RedeliveryPolicy, 3178
- redispatch
  - activemq::core::ActiveMQSession, 529
- ReentrantLock
  - decaf::util::concurrent::locks::ReentrantLock, 3183
- ReferenceType
  - decaf::lang::ArrayPointer, 732
  - decaf::lang::Pointer, 2948
- registerFactory
  - activemq::transport::TransportRegistry, 3903
  - activemq::wireformat::WireFormatRegistry, 4018
- rejectedExecution
  - decaf::util::concurrent::RejectedExecutionHandler, 3192
- RejectedExecutionException
  - decaf::util::concurrent::RejectedExecutionException, 3189, 3190
- relativize
  - decaf::net::URI, 3929
- release
  - decaf::lang::ArrayPointer, 735
  - decaf::lang::Pointer, 2951
  - decaf::util::concurrent::atomic::AtomicRefCounter, 747
  - decaf::util::concurrent::Semaphore, 3346
- releaseAll
  - activemq::cmsutil::ResourceLifecycleManager, 3284
- remaining
  - decaf::nio::Buffer, 932
- remainingCapacity
  - decaf::util::concurrent::BlockingQueue, 848
  - decaf::util::concurrent::SynchronousQueue, 3723
- remove
  - activemq::util::ActiveMQProperties, 481
  - cms::CMSProperties, 1167
  - decaf::internal::util::TimerTaskHeap, 3806
  - decaf::util::AbstractCollection, 187
  - decaf::util::AbstractQueue, 196
  - decaf::util::Collection, 1190
  - decaf::util::concurrent::ConcurrentMap, 1229
  - decaf::util::concurrent::ConcurrentStlMap, 1243, 1244
  - decaf::util::concurrent::SynchronousQueue, 3723
  - decaf::util::Iterator, 2155
  - decaf::util::List, 2342
  - decaf::util::Map, 2468
  - decaf::util::PriorityQueue, 3033, 3034
  - decaf::util::Properties, 3132
  - decaf::util::Queue, 3151
  - decaf::util::StlList, 3599, 3600
  - decaf::util::StlMap, 3611
  - decaf::util::StlSet, 3628
- removeAll
  - activemq::core::MessageDispatchChannel, 2603
  - decaf::util::AbstractCollection, 188
  - decaf::util::AbstractSet, 199
  - decaf::util::Collection, 1191
  - decaf::util::concurrent::SynchronousQueue, 3724
- removeConsumer
  - activemq::core::ActiveMQSession, 530
  - activemq::state::SessionState, 3438
- removeDispatcher
  - activemq::core::ActiveMQConnection, 286
- removeHandler
  - decaf::util::logging::Logger, 2395
- RemoveInfo
  - activemq::commands::RemoveInfo, 3195
- RemoveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3211
  - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3199
  - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3207
  - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3219
  - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3215
  - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3203
- removeProducer
  - activemq::core::ActiveMQConnection, 286
  - activemq::core::ActiveMQSession, 530
  - activemq::state::SessionState, 3438
- removeProperty
  - activemq::wireformat::stomp::StompFrame, 3636
- removePropertyChangeListener
  - decaf::util::logging::LogManager, 2411
- removeSession
  - activemq::core::ActiveMQConnection, 286
  - activemq::state::ConnectionState, 1391
- RemoveSubscriptionInfo

- activemq::commands::RemoveSubscriptionInfo, 3223
- RemoveSubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3228
  - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3236
  - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3232
  - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3248
  - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3244
  - activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3240
- removeSynchronization
  - activemq::core::ActiveMQTransactionContext, 721
- removeTask
  - activemq::threads::CompositeTaskRunner, 1224
- removeTempDestination
  - activemq::state::ConnectionState, 1391
- RemoveTransactionAction
  - activemq::state::ConnectionStateTracker, 1398
- removeTransactionState
  - activemq::state::ConnectionState, 1391
- removeTransportListener
  - activemq::core::ActiveMQConnection, 286
- removeURI
  - activemq::transport::CompositeTransport, 1226
  - activemq::transport::failover::FailoverTransport, 1880
  - activemq::transport::failover::URIPool, 3943
- renegotiateWireFormat
  - activemq::wireformat::openwire::OpenWireFormat, 2894
- replace
  - decaf::util::concurrent::ConcurrentMap, 1230, 1231
  - decaf::util::concurrent::ConcurrentStlMap, 1244, 1245
- ReplayCommand
  - activemq::commands::ReplayCommand, 3252
- ReplayCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3259
  - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3263
- activemq::wireformat::openwire::marshal::v3::ReplayCommand, 3267
- activemq::wireformat::openwire::marshal::v4::ReplayCommand, 3255
- activemq::wireformat::openwire::marshal::v5::ReplayCommand, 3275
- activemq::wireformat::openwire::marshal::v6::ReplayCommand, 3271
- replyTo
  - activemq::commands::ReplayCommand, 2532
- reportError
  - decaf::util::logging::Handler, 1980
- request
  - activemq::transport::correlator::ResponseCorrelator, 3293
- activemq::transport::failover::FailoverTransport, 1881
- activemq::transport::IOTransport, 2149
- activemq::transport::logging::LoggingTransport, 2404
- activemq::transport::mock::MockTransport, 2774
- activemq::transport::Transport, 3886, 3887
- activemq::transport::TransportFilter, 3896, 3897
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2903
- reserve
  - decaf::util::concurrent::ThreadPool, 3781
- reserved
  - z\_stream\_s, 4062
- reset
  - activemq::commands::ActiveMQBytesMessage, 243
  - activemq::commands::ActiveMQStreamMessage, 546
  - activemq::state::ConnectionState, 1391
  - cms::BytesMessage, 1065
  - decaf::internal::util::TimerTaskHeap, 3806
  - decaf::io::BufferedInputStream, 938
  - decaf::io::ByteArrayInputStream, 1025
  - decaf::io::ByteArrayOutputStream, 1029
  - decaf::io::FilterInputStream, 1898
  - decaf::io::InputStream, 2049
  - decaf::io::PushbackInputStream, 3144
  - decaf::io::Reader, 3168
  - decaf::lang::ArrayPointer, 735
  - decaf::lang::Pointer, 2952
  - decaf::nio::Buffer, 933
  - decaf::util::logging::LogManager, 2411
  - decaf::util::StlMapShrinker, 3670
  - decaf::util::zip::Adler32, 723
  - decaf::util::zip::CRC32, 1143
  - decaf::util::zip::CRC32, 1525

- decaf::util::zip::Deflater, 1711
- decaf::util::zip::Inflater, 2032
- decaf::util::zip::InflaterInputStream, 2040
- resize
  - decaf::internal::util::ByteArrayAdapter, 989
- resolve
  - decaf::net::URI, 3930
- resolveDestinationName
  - activemq::cmsutil::CmsDestinationAccessor, 1159
  - activemq::cmsutil::DestinationResolver, 1757
  - activemq::cmsutil::DynamicDestinationResolver, 1822
- ResolveProducerExecutor
  - activemq::cmsutil::CmsTemplate, 1182
  - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3278
- ResolveReceiveExecutor
  - activemq::cmsutil::CmsTemplate, 1182
  - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3279
- ResourceLifecycleManager
  - activemq::cmsutil::ResourceLifecycleManager, 3283
  - decaf::internal::util::ResourceLifecycleManager, 3281
- Response
  - activemq::commands::Response, 3286
- ResponseCorrelator
  - activemq::transport::correlator::ResponseCorrelator, 3292
- ResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3316
  - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3301
  - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3311
  - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3296
  - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3306
  - activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3321
- restore
  - activemq::state::ConnectionStateTracker, 1397
- restoreTransport
  - activemq::transport::failover::FailoverTransport, 1882
- result
  - activemq::commands::IntegerResponse, 2093
  - resume
    - activemq::commands::ConnectionControl, 1271
  - retainAll
    - decaf::util::AbstractCollection, 188
    - decaf::util::Collection, 1191
    - decaf::util::concurrent::SynchronousQueue, 3724
  - retroactive
    - activemq::commands::ConsumerInfo, 1466
  - returnInstance
    - decaf::util::logging::LogWriter, 2419
  - returnSession
    - activemq::cmsutil::SessionPool, 3436
  - reverse
    - decaf::lang::Integer, 2086
    - decaf::lang::Long, 2428
    - decaf::util::StlQueue, 3620
  - reverseBytes
    - decaf::lang::Integer, 2086
    - decaf::lang::Long, 2429
    - decaf::lang::Short, 3446
  - rewind
    - decaf::nio::Buffer, 933
  - rollback
    - activemq::cmsutil::PooledSession, 2966
    - activemq::core::ActiveMQConsumer, 317
    - activemq::core::ActiveMQSession, 530
    - activemq::core::ActiveMQTransactionContext, 721
  - rotateLeft
    - decaf::lang::Integer, 2086
    - decaf::lang::Long, 2429
  - rotateRight
    - decaf::lang::Integer, 2086
    - decaf::lang::Long, 2429
  - round
    - decaf::lang::Math, 2506
  - run
    - activemq::threads::CompositeTaskRunner, 1241
    - activemq::threads::DedicatedTaskRunner, 1241
  - activemq::transport::inactivity::ReadChecker, 3162
    - activemq::transport::inactivity::WriteChecker, 4020
    - activemq::transport::IOTransport, 2149
    - activemq::transport::mock::InternalCommandListener, 2123
    - decaf::lang::Runnable, 3325



- decaf::lang::Thread, 3771
- decaf::util::concurrent::PooledThread, 2969
- RUNNABLE
  - decaf::lang::Thread, 3768
- RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3328, 3329
- sane
  - inflate\_state, 2025
- schedule
  - decaf::util::Timer, 3792–3796
- scheduleAtFixedRate
  - decaf::util::Timer, 3797–3799
- scheduledExecutionTime
  - decaf::util::TimerTask, 3802
- SECONDS
  - decaf::util::concurrent::TimeUnit, 3815
- SecureRandom
  - decaf::security::SecureRandom, 3332, 3333
- SecureRandomImpl
  - decaf::internal::security::SecureRandomImpl, 3337
- SecureRandomSpi
  - decaf::security::SecureRandomSpi, 3339
- seek
  - gz\_state, 1977
- SEEK\_CUR
  - zconf.h, 4738
- SEEK\_END
  - zconf.h, 4738
- SEEK\_SET
  - zconf.h, 4738
- selector
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3176
  - activemq::commands::ConsumerInfo, 1466
  - activemq::commands::SubscriptionInfo, 3675
- Semaphore
  - decaf::util::concurrent::Semaphore, 3343
- semaphore
  - decaf::util::concurrent::ConditionHandle, 1255
- SEND
  - activemq::wireformat::stomp::StompCommand::ConsumerInfo, 3631
- send
  - activemq::cmsutil::CachedProducer, 1078, 1079
  - activemq::cmsutil::CmsTemplate, 1178, 1179
  - activemq::core::ActiveMQProducer, 474, 475
  - activemq::core::ActiveMQSession, 530
  - cms::MessageProducer, 2728, 2729
- SendExecutor
  - activemq::cmsutil::CmsTemplate, 1182
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3350
- sendPullRequest
  - activemq::core::ActiveMQConnection, 286
- sendUrgentData
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2866
  - decaf::net::Socket, 3514
  - decaf::net::SocketImpl, 3535
- ServerSocket
  - decaf::net::ServerSocket, 3354, 3355
  - decaf::net::Socket, 3518
- ServerSocketFactory
  - decaf::net::ServerSocketFactory, 3362
- serviceName
  - activemq::commands::DiscoveryEvent, 1761
- SESSION\_TRANSACTED
  - cms::Session, 3368
- SessionId
  - activemq::commands::SessionId, 3380
- sessionId
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionInfo, 3410
- SessionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3404
  - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3384
  - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3400
  - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3388
  - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3396
  - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3392
- SessionInfo
  - activemq::commands::SessionInfo, 3408
- SessionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3420
  - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3428
  - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3424
  - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3432

- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3781
- 3416
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3412
- 3636
- SessionPool
  - activemq::cmsutil::SessionPool, 3435
- SessionState
  - activemq::state::SessionState, 3438
- set
  - decaf::util::concurrent::atomic::AtomicBoolean, 739
  - decaf::util::concurrent::atomic::AtomicInteger, 745
  - decaf::util::concurrent::atomic::AtomicReference, 750
  - decaf::util::List, 2342
  - decaf::util::ListIterator, 2346
  - decaf::util::StlList, 3600
- setAbsolute
  - decaf::internal::net::URIType, 3956
- setAckHandler
  - activemq::commands::Message, 2527
- setAckMode
  - activemq::commands::SessionInfo, 3409
- setAckType
  - activemq::commands::MessageAck, 2562
- setAdditionalPredicate
  - activemq::commands::ConsumerInfo, 1464
- setAdvisory
  - activemq::commands::ActiveMQDestination, 329
- setAlwaysSyncSend
  - activemq::core::ActiveMQConnection, 287
  - activemq::core::ActiveMQConnectionFactory, 299
- setArrival
  - activemq::commands::Message, 2528
- setAuthority
  - decaf::internal::net::URIType, 3956
- setBackOffMultiplier
  - activemq::core::policies::DefaultRedeliveryPolicy, 1679
  - activemq::core::RedeliveryPolicy, 3180
  - activemq::transport::failover::FailoverTransport, 1882
- setBackup
  - activemq::transport::failover::FailoverTransport, 1882
- setBackupPoolSize
  - activemq::transport::failover::BackupTransport, 757
  - activemq::transport::failover::FailoverTransport, 1882
- setBlockSize
  - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3781
  - 3416
  - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3412
  - 3636
  - setBody
    - activemq::commands::ActiveMQBytesMessage, 243
  - setBodyBytes
    - cms::BytesMessage, 1065
  - setBool
    - activemq::util::PrimitiveList, 2986
    - activemq::util::PrimitiveMap, 2997
    - activemq::util::PrimitiveValueNode, 3022
  - setBoolean
    - activemq::commands::ActiveMQMapMessage, 369
    - cms::MapMessage, 2478
  - setBooleanProperty
    - activemq::commands::ActiveMQMessageTemplate, 436
    - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2736
    - cms::Message, 2549
  - setBranchQualifier
    - activemq::commands::XATransactionId, 4034
  - setBrokerId
    - activemq::commands::BrokerInfo, 901
  - setBrokerInTime
    - activemq::commands::Message, 2528
  - setBrokerMasterConnector
    - activemq::commands::ConnectionInfo, 1359
  - setBrokerName
    - activemq::commands::BrokerInfo, 901
    - activemq::commands::DiscoveryEvent, 1760
  - setBrokerOutTime
    - activemq::commands::Message, 2528
  - setBrokerPath
    - activemq::commands::ConnectionInfo, 1359
    - activemq::commands::ConsumerInfo, 1464
    - activemq::commands::DestinationInfo, 1730
    - activemq::commands::Message, 2528
    - activemq::commands::ProducerInfo, 3098
  - setBrokerSequenceId
    - activemq::commands::MessageId, 2666
  - setBrokerUploadUrl
    - activemq::commands::BrokerInfo, 901
  - setBrokerURL
    - activemq::commands::BrokerInfo, 901
    - activemq::core::ActiveMQConnection, 287

- activemq::core::ActiveMQConnectionFactory, 299
- setBrowser
  - activemq::commands::ConsumerInfo, 1464
- setByte
  - activemq::commands::ActiveMQMapMessage, 369
  - activemq::util::PrimitiveList, 2987
  - activemq::util::PrimitiveMap, 2998
  - activemq::util::PrimitiveValueNode, 3022
  - cms::MapMessage, 2478
- setByteArray
  - activemq::util::PrimitiveList, 2987
  - activemq::util::PrimitiveMap, 2998
  - activemq::util::PrimitiveValueNode, 3022
  - decaf::io::BlockingByteArrayInputStream, 841
  - decaf::io::ByteArrayInputStream, 1025, 1026
- setByteProperty
  - activemq::commands::ActiveMQMessageTemplate, 436
  - activemq::wireformat::openwire::utils::MessageProperty, 2736
  - cms::Message, 2550
- setBytes
  - activemq::commands::ActiveMQMapMessage, 369
  - cms::MapMessage, 2479
- setCacheEnabled
  - activemq::commands::WireFormatInfo, 3989
  - activemq::wireformat::openwire::OpenWireFormat, 2895
- setCacheSize
  - activemq::commands::WireFormatInfo, 3989
  - activemq::wireformat::openwire::OpenWireFormat, 2895
- setCause
  - activemq::commands::BrokerError, 865
- setChar
  - activemq::commands::ActiveMQMapMessage, 370
  - activemq::util::PrimitiveList, 2987
  - activemq::util::PrimitiveMap, 2998
  - activemq::util::PrimitiveValueNode, 3023
  - cms::MapMessage, 2479
- setCipherSuites
  - decaf::net::ssl::SSLParameters, 3552
- setClientID
  - activemq::core::ActiveMQConnection, 287
  - cms::Connection, 1265
- setClientId
  - activemq::core::ActiveMQConnectionFactory, 1359
  - activemq::commands::JournalTopicAck, 2186
  - activemq::commands::RemoveSubscriptionInfo, 3224
  - activemq::commands::SubscriptionInfo, 3674
  - activemq::core::ActiveMQConnectionFactory, 299
- setClientMaster
  - activemq::commands::ConnectionInfo, 1359
- setClose
  - activemq::commands::ConnectionControl, 1270
  - activemq::commands::ConsumerControl, 1404
- setClosed
  - activemq::transport::failover::BackupTransport, 753
- setCloseTimeout
  - activemq::core::ActiveMQConnection, 288
  - activemq::core::ActiveMQConnectionFactory, 299
- setCluster
  - activemq::commands::Message, 2528
- setCMSCorrelationID
  - activemq::commands::ActiveMQMessageTemplate, 437
  - cms::Message, 2550
- setCMSDeliveryMode
  - activemq::commands::ActiveMQMessageTemplate, 437
  - cms::Message, 2551
- setCMSDestination
  - activemq::commands::ActiveMQMessageTemplate, 437
  - cms::Message, 2551
- setCMSExpiration
  - activemq::commands::ActiveMQMessageTemplate, 437
  - cms::Message, 2552
- setCMSMessageID
  - activemq::commands::ActiveMQMessageTemplate, 438
  - cms::Message, 2552
- setCMSPriority
  - activemq::commands::ActiveMQMessageTemplate, 438
  - cms::Message, 2552
- setCMSRedelivered
  - activemq::commands::ActiveMQMessageTemplate, 438

- cms::Message, 2553
- setCMSReplyTo
  - activemq::commands::ActiveMQMessageTemplate, 3849
  - 439
  - cms::Message, 2553
- setCMSTimestamp
  - activemq::commands::ActiveMQMessageTemplate, 1882
  - 439
  - cms::Message, 2554
- setCMSType
  - activemq::commands::ActiveMQMessageTemplate, 439
  - cms::Message, 2554
- setCollisionAvoidancePercent
  - activemq::core::policies::DefaultRedeliveryPolicy, 1680
  - activemq::core::RedeliveryPolicy, 3180
- setCommand
  - activemq::commands::ControlCommand, 1495
  - activemq::wireformat::stomp::StompFrame, 3636
- setCommandId
  - activemq::commands::BaseCommand, 763
  - activemq::commands::Command, 1197
  - activemq::commands::PartialCommand, 2920
- setComponents
  - activemq::util::CompositeData, 1220
- setCompressed
  - activemq::commands::Message, 2528
- setConnectedBrokers
  - activemq::commands::ConnectionControl, 1270
- setConnection
  - activemq::commands::ActiveMQTempDestination, 578
  - activemq::commands::Message, 2528
- setConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1156
- setConnectionId
  - activemq::commands::BrokerInfo, 901
  - activemq::commands::ConnectionError, 1298
  - activemq::commands::ConnectionInfo, 1359
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::DestinationInfo, 1730
  - activemq::commands::LocalTransactionId, 2350
  - activemq::commands::ProducerId, 3070
  - activemq::commands::RemoveSubscriptionInfo, 3225
- activemq::commands::SessionId, 3382
- activemq::commands::TransactionInfo, 3849
- setConnectionInterruptProcessingComplete
  - activemq::state::ConnectionState, 1391
- activemq::transport::failover::FailoverTransport, 1882
- setConsumerId
  - activemq::commands::ConsumerControl, 1404
  - activemq::commands::ConsumerInfo, 1464
  - activemq::commands::MessageAck, 2563
  - activemq::commands::MessageDispatch, 2596
  - activemq::commands::MessageDispatchNotification, 2632
  - activemq::commands::MessagePull, 2742
- setContent
  - activemq::commands::Message, 2528
- setCorrelationId
  - activemq::commands::Message, 2529
  - activemq::commands::MessagePull, 2742
  - activemq::commands::Response, 3287
- setData
  - activemq::commands::DataArrayResponse, 1530
  - activemq::commands::DataResponse, 1585
  - activemq::commands::PartialCommand, 2920
- setDataStructure
  - activemq::commands::Message, 2529
- setDefault
  - decaf::net::ssl::SSLContext, 3547
- setDefaultClientId
  - activemq::core::ActiveMQConnection, 288
- setDefaultDestination
  - activemq::cmsutil::CmsTemplate, 1179
- setDefaultDestinationName
  - activemq::cmsutil::CmsTemplate, 1179
- setDeletedByBroker
  - activemq::commands::ActiveMQBlobMessage, 207
- setDeliveryMode
  - activemq::cmsutil::CachedProducer, 1080
  - activemq::cmsutil::CmsTemplate, 1180
  - activemq::core::ActiveMQProducer, 476
  - cms::MessageProducer, 2730
- setDeliveryPersistent
  - activemq::cmsutil::CmsTemplate, 1180
- setDeliverySequenceId
  - activemq::commands::MessageDispatchNotification, 2633
- setDestination

- activemq::commands::ConsumerControl, 1404
- activemq::commands::ConsumerInfo, 1464
- activemq::commands::DestinationInfo, 1730
- activemq::commands::JournalQueueAck, 2158
- activemq::commands::JournalTopicAck, 2186
- activemq::commands::Message, 2529
- activemq::commands::MessageAck, 2563
- activemq::commands::MessageDispatch, 2597
- activemq::commands::MessageDispatchNotification, 2633
- activemq::commands::MessagePull, 2742
- activemq::commands::ProducerInfo, 3099
- activemq::commands::SubscriptionInfo, 3674
- setDestinationResolver
  - activemq::cmsutil::CmsDestinationAccessor, 1159
- setDictionary
  - decaf::util::zip::Deflater, 1711, 1712
  - decaf::util::zip::Inflater, 2032, 2033
- setDisableMessageID
  - activemq::cmsutil::CachedProducer, 1080
  - activemq::core::ActiveMQProducer, 476
  - cms::MessageProducer, 2730
- setDisableMessageTimeStamp
  - activemq::cmsutil::CachedProducer, 1080
  - activemq::core::ActiveMQProducer, 476
  - cms::MessageProducer, 2730
- setDispatchAsync
  - activemq::commands::ConsumerInfo, 1464
  - activemq::commands::ProducerInfo, 3099
  - activemq::core::ActiveMQConnection, 288
  - activemq::core::ActiveMQConnectionFactory, 300
- setDouble
  - activemq::commands::ActiveMQMapMessage, 370
  - activemq::util::PrimitiveList, 2988
  - activemq::util::PrimitiveMap, 2998
  - activemq::util::PrimitiveValueNode, 3023
  - cms::MapMessage, 2479
- setDoubleProperty
  - activemq::commands::ActiveMQMessageTemplate, 439
  - activemq::wireformat::openwire::utils::MessageProperty, 2736
  - cms::Message, 2555
- setDroppable
  - activemq::commands::Message, 2529
- setDuplexConnection
  - activemq::commands::BrokerInfo, 901
- setDurableTopicPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1675
  - activemq::core::PrefetchPolicy, 2978
- setEnabled
  - activemq::transport::failover::BackupTransportPool, 757
- setEnabledCipherSuites
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2851
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2867
  - decaf::net::ssl::SSLServerSocket, 3558
  - decaf::net::ssl::SSLSocket, 3568
- setEnabledProtocols
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2851
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2867
  - decaf::net::ssl::SSLServerSocket, 3558
  - decaf::net::ssl::SSLSocket, 3568
- setenv
  - decaf::lang::System, 3732
- setErrorManager
  - decaf::util::logging::Handler, 1980
- setException
  - activemq::commands::ConnectionError, 1298
  - activemq::commands::ExceptionResponse, 1841
- setExceptionClass
  - activemq::commands::BrokerError, 866
- setExceptionListener
  - activemq::core::ActiveMQConnection, 288
  - activemq::core::ActiveMQConnectionFactory, 300
  - cms::Connection, 1265
- setExclusive
  - activemq::commands::ActiveMQDestination, 329
  - activemq::commands::ConsumerInfo, 1464
- setExit
  - activemq::commands::ConnectionControl, 1270
- setInterceptor
  - activemq::commands::Message, 2529
- setExpiration
  - activemq::commands::Message, 2529
- setExplicitQosEnabled
  - activemq::cmsutil::CmsTemplate, 1180

- setFailOnClose
  - activemq::transport::mock::MockTransport, 2775
- setFailOnKeepAliveSends
  - activemq::transport::mock::MockTransport, 2776
- setFailOnReceiveMessage
  - activemq::transport::mock::MockTransport, 2776
- setFailOnSendMessage
  - activemq::transport::mock::MockTransport, 2776
- setFailOnStart
  - activemq::transport::mock::MockTransport, 2776
- setFailOnStop
  - activemq::transport::mock::MockTransport, 2776
- setFaultTolerant
  - activemq::commands::ConnectionControl, 1270
  - activemq::commands::ConnectionInfo, 1359
- setFaultTolerantConfiguration
  - activemq::commands::BrokerInfo, 901
- setFilter
  - decaf::util::logging::Handler, 1980
  - decaf::util::logging::Logger, 2395
- setFirstMessageId
  - activemq::commands::MessageAck, 2563
- setFirstNakNumber
  - activemq::commands::ReplayCommand, 3253
- setFloat
  - activemq::commands::ActiveMQMapMessage, 370
  - activemq::util::PrimitiveList, 2988
  - activemq::util::PrimitiveMap, 2999
  - activemq::util::PrimitiveValueNode, 3023
  - cms::MapMessage, 2480
- setFloatProperty
  - activemq::commands::ActiveMQMessageTemplate, 440
  - activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2736
  - cms::Message, 2555
- setFlush
  - activemq::commands::ConsumerControl, 1404
- setFormatId
  - activemq::commands::XATransactionId, 4034
- setFormatter
  - decaf::util::logging::Handler, 1981
- setFragment
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3956
- setGlobalTransactionId
  - activemq::commands::XATransactionId, 4034
- setGroupID
  - activemq::commands::Message, 2529
- setGroupSequence
  - activemq::commands::Message, 2529
- setHost
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3957
- setInitialDelayTime
  - activemq::transport::inactivity::InactivityMonitor, 2006
- setInitialized
  - activemq::transport::failover::FailoverTransport, 1882
- setInitialReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1882
- setInitialRedeliveryDelay
  - activemq::core::policies::DefaultRedeliveryPolicy, 1680
  - activemq::core::RedeliveryPolicy, 3180
- setInput
  - decaf::util::zip::Deflater, 1712, 1713
  - decaf::util::zip::Inflater, 2033, 2034
- setInputStream
  - activemq::transport::IOTransport, 2150
- setInt
  - activemq::commands::ActiveMQMapMessage, 371
  - activemq::util::PrimitiveList, 2988
  - activemq::util::PrimitiveMap, 2999
  - activemq::util::PrimitiveValueNode, 3023
  - cms::MapMessage, 2480
- setIntProperty
  - activemq::commands::ActiveMQMessageTemplate, 440
  - activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2737
- setKeepAlive
  - decaf::net::Socket, 3514
- setKeepAliveResponseRequired
  - activemq::transport::inactivity::InactivityMonitor, 2007
- setLastDeliveredSequenceId
  - activemq::commands::RemoveInfo, 3196
  - activemq::core::ActiveMQConsumer, 318
  - activemq::core::ActiveMQSession, 531
- setLastMessageId
  - activemq::commands::MessageAck, 2563

- activemq::commands::MessageAck, 2563
- setLastNakNumber
  - activemq::commands::ReplayCommand, 3253
- setLevel
  - decaf::util::logging::Handler, 1981
  - decaf::util::logging::Logger, 2395
  - decaf::util::logging::LogRecord, 2416
  - decaf::util::zip::Deflater, 1713
- setLimit
  - activemq::util::MemoryUsage, 2514
- setList
  - activemq::util::PrimitiveValueNode, 3023
- setLoggerName
  - decaf::util::logging::LogRecord, 2416
- setLong
  - activemq::commands::ActiveMQMapMessage, 371
  - activemq::util::PrimitiveList, 2989
  - activemq::util::PrimitiveMap, 2999
  - activemq::util::PrimitiveValueNode, 3024
  - cms::MapMessage, 2480
- setLongProperty
  - activemq::commands::ActiveMQMessageTemplate, 440
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2737
  - cms::Message, 2556
- setMagic
  - activemq::commands::WireFormatInfo, 3989
- setManageable
  - activemq::commands::ConnectionInfo, 1359
- setManaged
  - decaf::internal::util::GenericResource, 1974
- setMap
  - activemq::util::PrimitiveValueNode, 3024
- setMark
  - cms::CMSException, 1162
  - decaf::lang::Exception, 1836
  - decaf::lang::Throwable, 3786
- setMarshaledForm
  - activemq::commands::BaseDataStructure, 832
  - activemq::wireformat::MarshalAware, 2486
- setMarshaledProperties
  - activemq::commands::Message, 2529
  - activemq::commands::WireFormatInfo, 3989
- setMasterBroker
  - activemq::commands::BrokerInfo, 901
- setMaxCacheSize
  - activemq::state::ConnectionStateTracker, 1398
  - activemq::transport::failover::FailoverTransport, 1883
- setMaximumPendingMessageLimit
  - activemq::commands::ConsumerInfo, 1464
- setMaximumRedeliveries
  - activemq::core::policies::DefaultRedeliveryPolicy, 1680
  - activemq::core::RedeliveryPolicy, 3180
- setMaxInactivityDuration
  - activemq::commands::WireFormatInfo, 3989
  - activemq::wireformat::openwire::OpenWireFormat, 2895
- setMaxInactivityDurationInitialDelay
  - activemq::commands::WireFormatInfo, 3989
  - activemq::wireformat::openwire::OpenWireFormat, 2895
- setMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1883
- setMaxReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1883
- setMaxThreads
  - decaf::util::concurrent::ThreadPool, 3781
- setMessage
  - activemq::commands::BrokerError, 866
  - activemq::commands::JournalTrace, 2214
  - activemq::commands::MessageDispatch, 2597
  - decaf::lang::Exception, 1836
  - decaf::util::logging::LogRecord, 2416
- setMessageAck
  - activemq::commands::JournalQueueAck, 2158
- setMessageCount
  - activemq::commands::MessageAck, 2563
- setMessageId
  - activemq::commands::JournalTopicAck, 2186
  - activemq::commands::Message, 2529
  - activemq::commands::MessageDispatchNotification, 2633
  - activemq::commands::MessagePull, 2742
- setMessageIdEnabled
  - activemq::cmsutil::CmsTemplate, 1180
- setMessageListener
  - activemq::cmsutil::CachedConsumer, 1073
  - activemq::core::ActiveMQConsumer, 318
  - cms::MessageConsumer, 2591

- setMessageSequenceId
  - activemq::commands::JournalTopicAck, 2186
- setMessageTimestampEnabled
  - activemq::cmsutil::CmsTemplate, 1181
- setMimeType
  - activemq::commands::ActiveMQBlobMessage, 207
- setName
  - activemq::commands::ActiveMQBlobMessage, 207
  - decaf::lang::Thread, 3771
- setNeedClientAuth
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2851
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2867
  - decaf::net::ssl::SSLParameters, 3553
  - decaf::net::ssl::SSLServerSocket, 3558
  - decaf::net::ssl::SSLSocket, 3569
- setNetworkBrokerId
  - activemq::commands::NetworkBridgeFilter, 2795
- setNetworkConnection
  - activemq::commands::BrokerInfo, 901
- setNetworkConsumerPath
  - activemq::commands::ConsumerInfo, 1464
- setNetworkProperties
  - activemq::commands::BrokerInfo, 901
- setNetworkSubscription
  - activemq::commands::ConsumerInfo, 1464
- setNetworkTTL
  - activemq::commands::NetworkBridgeFilter, 2795
- setNoLocal
  - activemq::cmsutil::CmsTemplate, 1181
  - activemq::commands::ConsumerInfo, 1464
- setNoRangeAcks
  - activemq::commands::ConsumerInfo, 1464
- setNumReceivedMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2776
- setNumReceivedMessages
  - activemq::transport::mock::MockTransport, 2776
- setNumSentKeepAlives
  - activemq::transport::mock::MockTransport, 2776
- setNumSentKeepAlivesBeforeFail
  - activemq::transport::mock::MockTransport, 2776
- setNumSentMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2776
- setNumSentMessages
  - activemq::transport::mock::MockTransport, 2776
- setObjectId
  - activemq::commands::RemoveInfo, 3196
- setOOBInline
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2868
  - decaf::net::Socket, 3514
- setOpaque
  - decaf::internal::net::URIType, 3957
- setOperationType
  - activemq::commands::DestinationInfo, 1730
- setOptimizedAcknowledge
  - activemq::commands::ConsumerInfo, 1464
- setOption
  - decaf::internal::net::tcp::TcpSocket, 3745
  - decaf::net::SocketImpl, 3535
- setOrdered
  - activemq::commands::ActiveMQDestination, 330
- setOrderedTarget
  - activemq::commands::ActiveMQDestination, 330
- setOriginalDestination
  - activemq::commands::Message, 2529
- setOriginalTransactionId
  - activemq::commands::Message, 2529
- setOutputStream
  - decaf::util::logging::StreamHandler, 3651
- setOutgoingListener
  - activemq::transport::mock::MockTransport, 2776
- setOutputStream
  - activemq::transport::IOTransport, 2150
- setParameters
  - activemq::util::CompositeData, 1220
- setParent
  - decaf::util::logging::Logger, 2395
- setPassword
  - activemq::commands::ConnectionInfo, 1359
  - activemq::core::ActiveMQConnection, 288
  - activemq::core::ActiveMQConnectionFactory, 300
- setPath
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3957
- setPeerBrokerInfos
  - activemq::commands::BrokerInfo, 901
- setPersistent



- activemq::commands::Message, 2529
- setPhysicalName
  - activemq::commands::ActiveMQDestination, 330
- setPooledThreadListener
  - decaf::util::concurrent::PooledThread, 2969
- setPort
  - decaf::internal::net::URIType, 3957
- setPreferedWireFormatInfo
  - activemq::wireformat::openwire::OpenWireFormatQuery, 2895
- setPrefetch
  - activemq::commands::ConsumerControl, 1404
- setPrefetchPolicy
  - activemq::core::ActiveMQConnection, 288
  - activemq::core::ActiveMQConnectionFactory, 300
- setPrefetchSize
  - activemq::commands::ConsumerInfo, 1464
- setPrepared
  - activemq::state::TransactionState, 3876
- setPreparedResult
  - activemq::state::TransactionState, 3876
- setPriority
  - activemq::cmsutil::CachedProducer, 1080
  - activemq::cmsutil::CmsTemplate, 1181
  - activemq::commands::ConsumerInfo, 1464
  - activemq::commands::Message, 2529
  - activemq::core::ActiveMQProducer, 476
  - cms::MessageProducer, 2731
  - decaf::lang::Thread, 3771
- setProducerId
  - activemq::commands::Message, 2529
  - activemq::commands::MessageId, 2666
  - activemq::commands::ProducerAck, 3038
  - activemq::commands::ProducerInfo, 3099
- setProducerSequenceId
  - activemq::commands::MessageId, 2666
- setProducerSessionKey
  - activemq::commands::ProducerId, 3070
- setProducerWindowSize
  - activemq::core::ActiveMQConnection, 289
  - activemq::core::ActiveMQConnectionFactory, 300
- setProperties
  - activemq::commands::WireFormatInfo, 3990
  - activemq::util::ActiveMQProperties, 481
  - decaf::util::logging::LogManager, 2411
- setProperty
  - activemq::util::ActiveMQProperties, 481
  - activemq::wireformat::stomp::StompFrame, 3637
  - cms::CMSProperties, 1167
  - decaf::lang::System, 3732
  - decaf::util::Properties, 3132
  - setProtocols
    - decaf::net::ssl::SSLParameters, 3553
  - setPubSubDomain
    - activemq::cmsutil::CmsDestinationAccessor, 1159
    - activemq::cmsutil::CmsTemplate, 1181
  - setQueue
    - decaf::internal::net::URIType, 3957
  - setQueueBrowserPrefetch
    - activemq::core::policies::DefaultPrefetchPolicy, 1675
    - activemq::core::PrefetchPolicy, 2979
  - setQueuePrefetch
    - activemq::core::policies::DefaultPrefetchPolicy, 1675
    - activemq::core::PrefetchPolicy, 2979
  - setRandomize
    - activemq::transport::failover::FailoverTransport, 1883
    - activemq::transport::failover::URIPool, 3944
  - setReadCheckTime
    - activemq::transport::inactivity::InactivityMonitor, 2007
  - setReadOnly
    - decaf::internal::nio::ByteBuffer, 1019
    - decaf::internal::nio::CharArrayBuffer, 1117
    - decaf::internal::nio::DoubleArrayBuffer, 1807
    - decaf::internal::nio::FloatArrayBuffer, 1923
    - decaf::internal::nio::IntArrayBuffer, 2064
    - decaf::internal::nio::LongArrayBuffer, 2442
    - decaf::internal::nio::ShortArrayBuffer, 3457
  - setReadOnlyBody
    - activemq::commands::Message, 2529
  - setReadOnlyProperties
    - activemq::commands::Message, 2529
  - setRebalanceConnection
    - activemq::commands::ConnectionControl, 1270
  - setReceiveBufferSize
    - decaf::net::ServerSocket, 3358
    - decaf::net::Socket, 3515
  - setReceiveTimeout
    - activemq::cmsutil::CmsTemplate, 1181
  - setRecievedByDFBridge
    - activemq::commands::Message, 2530
  - setReconnectDelay
    - activemq::transport::failover::FailoverTransport, 1883

- setReconnectTo
  - activemq::commands::ConnectionControl, 1270
- setRedeliveryCounter
  - activemq::commands::Message, 2530
  - activemq::commands::MessageDispatch, 2597
- setRedeliveryPolicy
  - activemq::core::ActiveMQConnection, 289
  - activemq::core::ActiveMQConnectionFactory, 301
  - activemq::core::ActiveMQConsumer, 318
- setRemoteBlobUrl
  - activemq::commands::ActiveMQBlobMessage, 207
- setReplyTo
  - activemq::commands::Message, 2530
- setResponse
  - activemq::transport::correlator::FutureResponse, 1970
- setResponseBuilder
  - activemq::transport::mock::InternalCommandList, 2123
  - activemq::transport::mock::MockTransport, 2776
- setResponseRequired
  - activemq::commands::BaseCommand, 763
  - activemq::commands::Command, 1197
- setRestoreConsumers
  - activemq::state::ConnectionStateTracker, 1398
- setRestoreProducers
  - activemq::state::ConnectionStateTracker, 1398
- setRestoreSessions
  - activemq::state::ConnectionStateTracker, 1398
- setRestoreTransaction
  - activemq::state::ConnectionStateTracker, 1398
- setResult
  - activemq::commands::IntegerResponse, 2093
- setResume
  - activemq::commands::ConnectionControl, 1270
- setRetroactive
  - activemq::commands::ConsumerInfo, 1464
- setReuseAddress
  - decaf::net::ServerSocket, 3359
  - decaf::net::Socket, 3515
- setScheduledTime
  - decaf::util::TimerTask, 3802
- setScheme
  - activemq::util::CompositeData, 1220
  - decaf::internal::net::URIType, 3957
- setSchemeSpecificPart
  - decaf::internal::net::URIType, 3958
- setSeed
  - decaf::security::SecureRandom, 3334, 3335
  - decaf::util::Random, 3159
- setSelector
  - activemq::commands::ConsumerInfo, 1464
  - activemq::commands::SubscriptionInfo, 3674
- setSendBufferSize
  - decaf::net::Socket, 3515
- setSendTimeout
  - activemq::core::ActiveMQConnection, 289
  - activemq::core::ActiveMQConnectionFactory, 301
  - activemq::core::ActiveMQProducer, 477
- setServerAuthority
  - decaf::internal::net::URIType, 3958
- setServiceName
  - activemq::commands::DiscoveryEvent, 1760
- setSessionAcknowledgeMode
  - activemq::cmsutil::CmsAccessor, 1156
- setSessionId
  - activemq::commands::ConsumerId, 1433
  - activemq::commands::ProducerId, 3070
  - activemq::commands::SessionInfo, 3409
- setShort
  - activemq::commands::ActiveMQMapMessage, 371
  - activemq::util::PrimitiveList, 2989
  - activemq::util::PrimitiveMap, 2999
  - activemq::util::PrimitiveValueNode, 3024
  - cms::MapMessage, 2481
- setShortProperty
  - activemq::commands::ActiveMQMessageTemplate, 441
  - activemq::wireformat::openwire::utils::MessagePropertyInterface, 2737
  - cms::Message, 2557
- setSize
  - activemq::commands::ProducerAck, 3038
- setSizePrefixDisabled
  - activemq::commands::WireFormatInfo, 3990
  - activemq::wireformat::openwire::OpenWireFormat, 2896
- setSlaveBroker
  - activemq::commands::BrokerInfo, 901
- setSocketImplFactory
  - decaf::net::ServerSocket, 3359
  - decaf::net::Socket, 3516

- setSoLinger
  - decaf::net::Socket, 3516
- setSoTimeout
  - decaf::net::ServerSocket, 3359
  - decaf::net::Socket, 3516
- setSource
  - decaf::internal::net::URIType, 3958
- setSourceFile
  - decaf::util::logging::LogRecord, 2416
- setSourceFunction
  - decaf::util::logging::LogRecord, 2416
- setSourceLine
  - decaf::util::logging::LogRecord, 2416
- setSSLParameters
  - decaf::net::ssl::SSLSocket, 3569
- setStackTrace
  - decaf::lang::Exception, 1837
- setStackTraceElements
  - activemq::commands::BrokerError, 866
- setStackTraceEnabled
  - activemq::commands::WireFormatInfo, 3990
  - activemq::wireformat::openwire::OpenWireFormat, 2896
- setStart
  - activemq::commands::ConsumerControl, 1404
- setStartupMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1883
- setStop
  - activemq::commands::ConsumerControl, 1404
- setStrategy
  - decaf::util::zip::Deflater, 1714
- setString
  - activemq::commands::ActiveMQMapMessage, 372
  - activemq::util::PrimitiveList, 2989
  - activemq::util::PrimitiveMap, 2999
  - activemq::util::PrimitiveValueNode, 3024
  - cms::MapMessage, 2481
- setStringProperty
  - activemq::commands::ActiveMQMessageTemplate, 441
  - activemq::wireformat::openwire::utils::MessageProperty, 2737
  - cms::Message, 2557
- setSubscriptionName
  - activemq::commands::RemoveSubscriptionInfo, 3225
  - activemq::commands::SubscriptionInfo, 3674
- setSubscribedDestination
  - activemq::commands::SubscriptionInfo, 3674
- setSubscriptionName
  - activemq::commands::ConsumerInfo, 1464
- setSubscriptionName
  - activemq::commands::JournalTopicAck, 2186
- setSuspend
  - activemq::commands::ConnectionControl, 1270
- setSynchronizationRegistered
  - activemq::core::ActiveMQConsumer, 318
- setTargetConsumerId
  - activemq::commands::Message, 2530
- setTcpNoDelay
  - decaf::net::Socket, 3517
- setTcpNoDelayEnabled
  - activemq::commands::WireFormatInfo, 3990
  - activemq::wireformat::openwire::OpenWireFormat, 2896
- setText
  - activemq::commands::ActiveMQTextMessage, 665
  - cms::TextMessage, 3764
- setTextView
  - activemq::commands::MessageId, 2666
- setThrown
  - decaf::util::logging::LogRecord, 2417
- setTightEncodingEnabled
  - activemq::commands::WireFormatInfo, 3990
  - activemq::wireformat::openwire::OpenWireFormat, 2896
- setTime
  - decaf::util::Date, 1668
- setTimeout
  - activemq::commands::DestinationInfo, 1730
  - activemq::commands::MessagePull, 2742
  - activemq::transport::failover::FailoverTransport, 1883
- setTimestamp
  - activemq::commands::Message, 2530
  - decaf::util::logging::LogRecord, 2417
- setTopic
  - activemq::cmsutil::CachedProducer, 1081
  - activemq::cmsutil::CmsTemplate, 1181
  - activemq::core::ActiveMQProducer, 477
  - cms::MessageProducer, 2731
- setTopicPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1676
  - activemq::core::PrefetchPolicy, 2979

- setTrackMessages
  - activemq::state::ConnectionStateTracker, 1398
  - activemq::transport::failover::FailoverTransport, 1883
- setTrackTransactionProducers
  - activemq::state::ConnectionStateTracker, 1398
  - activemq::transport::failover::FailoverTransport, 1883
- setTrackTransactions
  - activemq::state::ConnectionStateTracker, 1398
- setTrafficClass
  - decaf::net::Socket, 3517
- setTransactionId
  - activemq::commands::JournalTopicAck, 2186
  - activemq::commands::JournalTransaction, 2241
  - activemq::commands::Message, 2530
  - activemq::commands::MessageAck, 2563
  - activemq::commands::TransactionInfo, 3849
- setTransactionState
  - activemq::state::ProducerState, 3125
- setTransport
  - activemq::transport::failover::BackupTransport, 753
  - activemq::transport::mock::InternalCommandList, 2123
- setTransportInterruptionProcessingComplete
  - activemq::core::ActiveMQConnection, 289
- setTransportListener
  - activemq::transport::failover::FailoverTransport, 1883
  - activemq::transport::IOTransport, 2150
  - activemq::transport::mock::MockTransport, 2777
  - activemq::transport::Transport, 3887
  - activemq::transport::TransportFilter, 3897
- setTreadId
  - decaf::util::logging::LogRecord, 2417
- setType
  - activemq::commands::JournalTransaction, 2241
  - activemq::commands::Message, 2530
  - activemq::commands::TransactionInfo, 3849
- setUncaughtExceptionHandler
  - decaf::lang::Thread, 3771
- setUpSocketImpl
  - decaf::net::ServerSocket, 3360
- setUri
  - activemq::transport::failover::BackupTransport, 753
  - setUsage
    - activemq::util::MemoryUsage, 2514
  - setUseAsyncSend
    - activemq::core::ActiveMQConnection, 289
    - activemq::core::ActiveMQConnectionFactory, 301
  - setUseClientMode
    - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2846
    - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2868
    - decaf::net::ssl::SSLSocket, 3569
  - setUseCollisionAvoidance
    - activemq::core::policies::DefaultRedeliveryPolicy, 1680
    - activemq::core::RedeliveryPolicy, 3181
  - setUseCompression
    - activemq::core::ActiveMQConnection, 290
    - activemq::core::ActiveMQConnectionFactory, 301
  - setUseExponentialBackOff
    - activemq::core::policies::DefaultRedeliveryPolicy, 1680
    - activemq::core::RedeliveryPolicy, 3181
    - activemq::transport::failover::FailoverTransport, 1883
  - setUseParentHandlers
    - decaf::util::logging::Logger, 2395
  - setUserID
    - activemq::commands::Message, 2530
  - setUserInfo
    - decaf::internal::net::URIType, 3958
  - setUserName
    - activemq::commands::ConnectionInfo, 1359
  - setUsername
    - activemq::core::ActiveMQConnection, 290
    - activemq::core::ActiveMQConnectionFactory, 301
  - setValid
    - decaf::internal::net::URIType, 3958
  - setValue
    - activemq::commands::BrokerId, 871
    - activemq::commands::ConnectionId, 1329
    - activemq::commands::ConsumerId, 1433
    - activemq::commands::LocalTransactionId, 2350
    - activemq::commands::MessageId, 2666
    - activemq::commands::ProducerId, 3070
    - activemq::commands::SessionId, 3382
    - activemq::util::PrimitiveValueNode, 3024
    - decaf::util::Map::Entry, 1824

- setVersion
  - activemq::commands::WireFormatInfo, 3990
  - activemq::wireformat::openwire::OpenWireFormat, 2896
  - activemq::wireformat::stomp::StompWireFormat, 3645
  - activemq::wireformat::WireFormat, 3978
- setWantClientAuth
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 1224
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2852
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2868
  - decaf::net::ssl::SSLParameters, 3553
  - decaf::net::ssl::SSLServerSocket, 3558
  - decaf::net::ssl::SSLSocket, 3570
- setWasPrepared
  - activemq::commands::JournalTransaction, 2241
- setWindowSize
  - activemq::commands::ProducerInfo, 3099
- setWireFormat
  - activemq::transport::failover::FailoverTransport, 1884
  - activemq::transport::IOTransport, 2150
  - activemq::transport::mock::MockTransport, 2777
  - activemq::transport::Transport, 3888
  - activemq::transport::TransportFilter, 3897
- setWriteCheckTime
  - activemq::transport::inactivity::InactivityMonitor, 2007
- SEVERE
  - decaf::util::logging::Level, 2336
- severe
  - decaf::util::logging::Logger, 2396
- Short
  - decaf::lang::Short, 3441
- SHORT\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- ShortArrayBuffer
  - decaf::internal::nio::ShortArrayBuffer, 3452, 3453
- ShortBuffer
  - decaf::nio::ShortBuffer, 3461
- shortValue
  - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 3009
  - decaf::lang::Byte, 966
  - decaf::lang::Character, 1106
  - decaf::lang::Double, 1796
  - decaf::lang::Float, 1912
  - decaf::lang::Integer, 2087
  - decaf::lang::Long, 2430
  - decaf::lang::Number, 2837
  - decaf::lang::Short, 3446
  - shutdown
    - activemq::state::ConnectionState, 1391
    - activemq::state::SessionState, 3438
    - activemq::state::TransactionState, 3876
    - activemq::threads::CompositeTaskRunner, 1671, 1672
    - activemq::threads::DedicatedTaskRunner, 3736
    - activemq::threads::TaskRunner, 3736
    - ShutdownInfo
      - activemq::commands::ShutdownInfo, 3471
    - ShutdownInfoMarshaller
      - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3482
      - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3478
      - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3490
      - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3494
      - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3486
      - activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3474
    - shutdownInput
      - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2869
      - decaf::internal::net::tcp::TcpSocket, 3745
      - decaf::net::Socket, 3517
      - decaf::net::SocketImpl, 3535
    - shutdownLibrary
      - activemq::library::ActiveMQCPP, 321
    - shutdownNetworking
      - decaf::internal::net::Network, 2791
    - shutdownOutput
      - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2869
      - decaf::internal::net::tcp::TcpSocket, 3745
      - decaf::net::Socket, 3517
      - decaf::net::SocketImpl, 3535
    - shutdownRuntime
      - decaf::lang::Runtime, 3327
    - signal
      - decaf::util::concurrent::locks::Condition, 1254
    - signalAll
      - decaf::util::concurrent::locks::Condition, 1254
  - SignatureException

- decaf::security::SignatureException, 3497, 3498
- signum
  - decaf::lang::Integer, 2087
  - decaf::lang::Long, 2430
  - decaf::lang::Math, 2506, 2507
- SimpleFormatter
  - decaf::util::logging::SimpleFormatter, 3500
- SimpleLogger
  - decaf::util::logging::SimpleLogger, 3501
- SIZE
  - decaf::lang::Byte, 968
  - decaf::lang::Character, 1107
  - decaf::lang::Double, 1798
  - decaf::lang::Float, 1914
  - decaf::lang::Integer, 2090
  - decaf::lang::Long, 2433
  - decaf::lang::Short, 3448
- size
  - activemq::commands::ProducerAck, 3039
  - activemq::core::MessageDispatchChannel, 2603
  - activemq::wireformat::openwire::utils::HexTable, 1985
  - decaf::internal::util::TimerTaskHeap, 3806
  - decaf::io::ByteArrayOutputStream, 1029
  - decaf::io::DataOutputStream, 1581
  - decaf::util::Collection, 1191
  - decaf::util::concurrent::ConcurrentStlMap, 1245
  - decaf::util::concurrent::SynchronousQueue, 3724
  - decaf::util::Map, 2469
  - decaf::util::PriorityQueue, 3034
  - decaf::util::Properties, 1132
  - decaf::util::StlList, 3601
  - decaf::util::StlMap, 3611
  - decaf::util::StlQueue, 3620
  - decaf::util::StlSet, 3628
  - gz\_state, 1977
- skip
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2884
  - decaf::internal::net::tcp::TcpSocketInputStream, 3749
  - decaf::io::BlockingByteArrayInputStream, 841
  - decaf::io::BufferedInputStream, 938
  - decaf::io::ByteArrayInputStream, 1026
  - decaf::io::FilterInputStream, 1899
  - decaf::io::InputStream, 2050
  - decaf::io::PushbackInputStream, 3145
  - decaf::io::Reader, 3168
  - decaf::util::zip::CheckedInputStream, 1138
  - decaf::util::zip::InflaterInputStream, 2040
  - gz\_state, 1977
- skipBytes
  - decaf::io::DataInput, 1565
  - decaf::io::DataInputStream, 1573
- slaveBroker
  - activemq::commands::BrokerInfo, 903
- sleep
  - decaf::lang::Thread, 3772
  - decaf::util::concurrent::TimeUnit, 3811
- SLEEPING
  - decaf::lang::Thread, 3768
- slice
  - decaf::internal::nio::ByteBuffer, 1019
  - decaf::internal::nio::CharArrayBuffer, 1117
  - decaf::internal::nio::DoubleArrayBuffer, 1807
  - decaf::internal::nio::FloatArrayBuffer, 1923
  - decaf::internal::nio::IntArrayBuffer, 2064
  - decaf::internal::nio::LongArrayBuffer, 2442
  - decaf::internal::nio::ShortArrayBuffer, 3457
  - decaf::nio::ByteBuffer, 1053
  - decaf::nio::CharBuffer, 1132
  - decaf::nio::DoubleBuffer, 1818
  - decaf::nio::FloatBuffer, 1933
  - decaf::nio::IntBuffer, 2075
  - decaf::nio::LongBuffer, 2453
  - decaf::nio::ShortBuffer, 3467
- Socket
  - decaf::net::Socket, 3506–3508
- SOCKET\_OPTION\_BINDADDR
  - decaf::net::SocketOptions, 3539
- SOCKET\_OPTION\_BROADCAST
  - decaf::net::SocketOptions, 3539
- SOCKET\_OPTION\_IP\_MULTICAST\_IF
  - decaf::net::SocketOptions, 3539
- SOCKET\_OPTION\_IP\_MULTICAST\_IF2
  - decaf::net::SocketOptions, 3539
- SOCKET\_OPTION\_IP\_MULTICAST\_LOOP
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_IP\_TOS
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_KEEPAIVE
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_LINGER
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_OOBLINER
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_RCVBUF
  - decaf::net::SocketOptions, 3540
- SOCKET\_OPTION\_REUSEADDR
  - decaf::net::SocketOptions, 3541

SOCKET_OPTION_SNDBUF	src/main/activemq/commands/ActiveMQMessage.h,
decaf::net::SocketOptions, 3541	4085
SOCKET_OPTION_TCP_NODELAY	src/main/activemq/commands/ActiveMQMessageTemplate.h,
decaf::net::SocketOptions, 3541	4086
SOCKET_OPTION_TIMEOUT	src/main/activemq/commands/ActiveMQObjectMessage.h,
decaf::net::SocketOptions, 3541	4087
SocketException	src/main/activemq/commands/ActiveMQQueue.h,
decaf::net::SocketException, 3521, 3522	4088
SocketFactory	src/main/activemq/commands/ActiveMQStreamMessage.h,
decaf::net::SocketFactory, 3525	4089
SocketFileDescriptor	src/main/activemq/commands/ActiveMQTempDestination.h,
decaf::internal::net::SocketFileDescriptor,	4090
3528	src/main/activemq/commands/ActiveMQTempQueue.h,
SocketImpl	4091
decaf::net::SocketImpl, 3531	src/main/activemq/commands/ActiveMQTempTopic.h,
SocketTimeoutException	4092
decaf::net::SocketTimeoutException, 3542,	src/main/activemq/commands/ActiveMQTextMessage.h,
3543	4093
sqrt	src/main/activemq/commands/ActiveMQTopic.h,
decaf::lang::Math, 2508	4094
src/main/activemq/cmsutil/CachedConsumer.h,	src/main/activemq/commands/BaseCommand.h,
4067	4095
src/main/activemq/cmsutil/CachedProducer.h,	src/main/activemq/commands/BaseDataStructure.h,
4068	4096
src/main/activemq/cmsutil/CmsAccessor.h,	src/main/activemq/commands/BooleanExpression.h,
4069	4097
src/main/activemq/cmsutil/CmsDestinationAccessControlList.h,	src/main/activemq/commands/BrokerError.h,
4070	4098
src/main/activemq/cmsutil/CmsTemplate.h,	src/main/activemq/commands/BrokerId.h,
4071	4099
src/main/activemq/cmsutil/DestinationResolver.h,	src/main/activemq/commands/BrokerInfo.h,
4072	4100
src/main/activemq/cmsutil/DynamicDestinationResolver.h,	src/main/activemq/commands/Command.h,
4073	4101
src/main/activemq/cmsutil/MessageCreator.h,	src/main/activemq/commands/ConnectionControl.h,
4074	4102
src/main/activemq/cmsutil/PooledSession.h,	src/main/activemq/commands/ConnectionError.h,
4075	4103
src/main/activemq/cmsutil/ProducerCallback.h,	src/main/activemq/commands/ConnectionId.h,
4076	4104
src/main/activemq/cmsutil/ResourceLifecycleManager.h,	src/main/activemq/commands/ConnectionInfo.h,
4077	4105
src/main/activemq/cmsutil/SessionCallback.h,	src/main/activemq/commands/ConsumerControl.h,
4079	4106
src/main/activemq/cmsutil/SessionPool.h,	src/main/activemq/commands/ConsumerId.h,
4080	4107
src/main/activemq/commands/ActiveMQBlobMessage.h,	src/main/activemq/commands/ConsumerInfo.h,
4081	4108
src/main/activemq/commands/ActiveMQBytesMessage.h,	src/main/activemq/commands/ControlCommand.h,
4082	4109
src/main/activemq/commands/ActiveMQDestination.h,	src/main/activemq/commands/DataArrayResponse.h,
4083	4110
src/main/activemq/commands/ActiveMQMapMessage.h,	src/main/activemq/commands/DataResponse.h,
4084	4111

src/main/activemq/commands/DataStructure.h, src/main/activemq/commands/Response.h,  
 4112 4140  
 src/main/activemq/commands/DestinationInfo.h,src/main/activemq/commands/SessionId.h,  
 4113 4141  
 src/main/activemq/commands/DiscoveryEvent.h,src/main/activemq/commands/SessionInfo.h,  
 4114 4142  
 src/main/activemq/commands/ExceptionResponse.h,src/main/activemq/commands/ShutdownInfo.h,  
 4115 4143  
 src/main/activemq/commands/FlushCommand.hsrc/main/activemq/commands/SubscriptionInfo.h,  
 4116 4144  
 src/main/activemq/commands/IntegerResponse.hsrc/main/activemq/commands/TransactionId.h,  
 4117 4145  
 src/main/activemq/commands/JournalQueueAck.hsrc/main/activemq/commands/TransactionInfo.h,  
 4118 4146  
 src/main/activemq/commands/JournalTopicAck.hsrc/main/activemq/commands/WireFormatInfo.h,  
 4119 4147  
 src/main/activemq/commands/JournalTrace.h, src/main/activemq/commands/XATransactionId.h,  
 4120 4148  
 src/main/activemq/commands/JournalTransaction.h,src/main/activemq/core/ActiveMQAckHandler.h,  
 4121 4149  
 src/main/activemq/commands/KeepAliveInfo.h, src/main/activemq/core/ActiveMQConnection.h,  
 4122 4150  
 src/main/activemq/commands/LastPartialCommand.h,src/main/activemq/core/ActiveMQConnectionFactory.h,  
 4123 4151  
 src/main/activemq/commands/LocalTransactionId.h,src/main/activemq/core/ActiveMQConnectionMetaData.h,  
 4124 4152  
 src/main/activemq/commands/Message.h, src/main/activemq/core/ActiveMQConstants.h,  
 4125 4153  
 src/main/activemq/commands/MessageAck.h, src/main/activemq/core/ActiveMQConsumer.h,  
 4127 4154  
 src/main/activemq/commands/MessageDispatch.hsrc/main/activemq/core/ActiveMQProducer.h,  
 4128 4155  
 src/main/activemq/commands/MessageDispatchNotification.h,src/main/activemq/core/ActiveMQQueueBrowser.h,  
 4129 4156  
 src/main/activemq/commands/MessageId.h, src/main/activemq/core/ActiveMQSession.h,  
 4130 4157  
 src/main/activemq/commands/MessagePull.h, src/main/activemq/core/ActiveMQSessionExecutor.h,  
 4131 4158  
 src/main/activemq/commands/NetworkBridgeFilter.h,src/main/activemq/core/ActiveMQTransactionContext.h,  
 4132 4159  
 src/main/activemq/commands/PartialCommand.hsrc/main/activemq/core/DispatchData.h, 4160  
 4133 src/main/activemq/core/Dispatcher.h, 4161  
 src/main/activemq/commands/ProducerAck.h, src/main/activemq/core/MessageDispatchChannel.h,  
 4134 4162  
 src/main/activemq/commands/ProducerId.h, src/main/activemq/core/policies/DefaultPrefetchPolicy.h,  
 4135 4163  
 src/main/activemq/commands/ProducerInfo.h, src/main/activemq/core/policies/DefaultRedeliveryPolicy.h,  
 4136 4164  
 src/main/activemq/commands/RemoveInfo.h, src/main/activemq/core/PrefetchPolicy.h, 4165  
 4137 src/main/activemq/core/RedeliveryPolicy.h,  
 src/main/activemq/commands/RemoveSubscriptionInfo.h, 4166  
 4138 src/main/activemq/core/Synchronization.h,  
 src/main/activemq/commands/ReplayCommand.h, 4167  
 4139 src/main/activemq/exceptions/ActiveMQException.h,



- 4168
- src/main/activemq/exceptions/BrokerException.h, 4202
- 4169
- src/main/activemq/exceptions/ExceptionDefines.h, 4203
- 4170
- src/main/activemq/io/LoggingInputStream.h, 4204
- 4175
- src/main/activemq/io/LoggingOutputStream.h, 4205
- 4176
- src/main/activemq/library/ActiveMQCPP.h, 4206
- 4177
- src/main/activemq/state/CommandVisitor.h, 4207
- 4178
- src/main/activemq/state/CommandVisitorAdapter.h, 4208
- 4179
- src/main/activemq/state/ConnectionState.h, 4209
- 4181
- src/main/activemq/state/ConnectionStateTracker.h, 4210
- 4182
- src/main/activemq/state/ConsumerState.h, 4211
- 4183
- src/main/activemq/state/ProducerState.h, 4212
- 4184
- src/main/activemq/state/SessionState.h, 4185
- src/main/activemq/state/Tracked.h, 4186
- src/main/activemq/state/TransactionState.h, 4187
- src/main/activemq/threads/CompositeTask.h, 4188
- src/main/activemq/threads/CompositeTaskRunner.h, 4189
- src/main/activemq/threads/DedicatedTaskRunner.h, 4190
- src/main/activemq/threads/Task.h, 4191
- src/main/activemq/threads/TaskRunner.h, 4192
- src/main/activemq/transport/AbstractTransportFactory.h, 4193
- src/main/activemq/transport/CompositeTransport.h, 4194
- src/main/activemq/transport/correlator/FutureResponse.h, 4195
- src/main/activemq/transport/correlator/ResponseCorrelator.h, 4196
- src/main/activemq/transport/DefaultTransportListener.h, 4197
- src/main/activemq/transport/failover/BackupTransport.h, 4198
- src/main/activemq/transport/failover/BackupTransportPool.h, 4199
- src/main/activemq/transport/failover/CloseTransport.h, 4200
- src/main/activemq/transport/failover/FailoverTransport.h, 4201
- src/main/activemq/transport/failover/FailoverTransportFactory.h, 4202
- src/main/activemq/transport/failover/FailoverTransportListener.h, 4203
- src/main/activemq/transport/failover/URIPool.h, 4204
- src/main/activemq/transport/inactivity/InactivityMonitor.h, 4205
- src/main/activemq/transport/inactivity/ReadChecker.h, 4206
- src/main/activemq/transport/inactivity/WriteChecker.h, 4207
- src/main/activemq/transport/IOTransport.h, 4208
- src/main/activemq/transport/logging/LoggingTransport.h, 4209
- src/main/activemq/transport/mock/InternalCommandListener.h, 4210
- src/main/activemq/transport/mock/MockTransport.h, 4211
- src/main/activemq/transport/mock/MockTransportFactory.h, 4212
- src/main/activemq/transport/mock/ResponseBuilder.h, 4213
- src/main/activemq/transport/tcp/SslTransport.h, 4214
- src/main/activemq/transport/tcp/SslTransportFactory.h, 4215
- src/main/activemq/transport/tcp/TcpTransport.h, 4216
- src/main/activemq/transport/tcp/TcpTransportFactory.h, 4217
- src/main/activemq/transport/Transport.h, 4218
- src/main/activemq/transport/TransportFactory.h, 4219
- src/main/activemq/transport/TransportFilter.h, 4220
- src/main/activemq/transport/TransportListener.h, 4221
- src/main/activemq/transport/TransportRegistry.h, 4222
- src/main/activemq/transport/correlator/ResponseCorrelator.h, 4223
- src/main/activemq/util/ActiveMQProperties.h, 4224
- src/main/activemq/util/CMSExceptionSupport.h, 4225
- src/main/activemq/util/CompositeData.h, 4226
- src/main/activemq/util/Config.h, 4227
- src/main/activemq/util/IdGenerator.h, 4230
- src/main/activemq/util/LongSequenceGenerator.h, 4231
- src/main/activemq/util/MarshallingSupport.h, 4232

- src/main/activemq/util/MemoryUsage.h, 4233
- src/main/activemq/util/PrimitiveList.h, 4234
- src/main/activemq/util/PrimitiveMap.h, 4235
- src/main/activemq/util/PrimitiveValueConverter.h, 4236
- src/main/activemq/util/PrimitiveValueNode.h, 4237
- src/main/activemq/util/URISupport.h, 4238
- src/main/activemq/util/Usage.h, 4239
- src/main/activemq/wireformat/MarshalAware.h, 4240
- src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h, 4241
- src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h, 4242
- src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h, 4243
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h, 4244
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h, 4250
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h, 4256
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h, 4262
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h, 4268
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h, 4274
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h, 4280
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h, 4286
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h, 4292
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h, 4298
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h, 4304
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h, 4310
- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h, 4316
- src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h, 4322
- src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h, 4328
- src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h, 4334
- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h, 4340
- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h, 4346
- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h, 4352
- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h, 4358
- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerConnectionIdMarshaller.h, 4364
- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h, 4370
- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h, 4376
- src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h, 4382
- src/main/activemq/wireformat/openwire/marshal/v1/DataArrayMarshaller.h, 4388
- src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h, 4394
- src/main/activemq/wireformat/openwire/marshal/v1/DestinationIdMarshaller.h, 4400
- src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryErrorMarshaller.h, 4406
- src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h, 4412
- src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h, 4418
- src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h, 4424
- src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueIdMarshaller.h, 4430
- src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicIdMarshaller.h, 4436
- src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceIdMarshaller.h, 4442
- src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionIdMarshaller.h, 4448
- src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h, 4454
- src/main/activemq/wireformat/openwire/marshal/v1/LastPartialMessageIdMarshaller.h, 4460
- src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h, 4466
- src/main/activemq/wireformat/openwire/marshal/v1/Marshallable.h, 4472
- src/main/activemq/wireformat/openwire/marshal/v1/MessageAcknowledgeIdMarshaller.h, 4478
- src/main/activemq/wireformat/openwire/marshal/v1/MessageDestinationIdMarshaller.h, 4484
- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h, 4490
- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h, 4496
- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h, 4502
- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h, 4508

src/main/activemq/wireformat/openwire/marshalsrv1/networkbridge/filter/marshall/openwire/marshal/v2/ActiveMQT	
4514	4305
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4520	4311
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4526	4317
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4532	4323
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4538	4329
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4544	4335
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4550	4341
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4556	4347
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4562	4353
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4568	4359
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4574	4365
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4580	4371
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4586	4377
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4592	4383
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4598	4389
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4604	4395
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4610	4401
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4616	4407
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4622	4413
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4628	4419
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4634	4425
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4640	4431
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4646	4437
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4652	4443
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4658	4449
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4664	4455
src/main/activemq/wireformat/openwire/marshalsrv1/partialcomm/marshall/openwire/marshal/v2/ActiveMQT	
4670	4461

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQD  
4467 4258

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQD  
4473 4264

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQD  
4479 4270

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQC  
4485 4276

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQC  
4491 4282

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQS  
4497 4288

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQT  
4503 4294

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQT  
4509 4300

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQT  
4515 4306

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQT  
4521 4312

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ActiveMQT  
4527 4318

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/BaseComm  
4533 4324

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/BrokerIdMa  
4539 4330

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/BrokerInfo  
4545 4336

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/Connection  
4551 4342

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/Connection  
4557 4348

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/Connection  
4563 4354

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/Connection  
4569 4360

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ConsumerC  
4575 4366

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ConsumerId  
4581 4372

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ConsumerIn  
4587 4378

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/ControlCom  
4593 4384

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/DataArrayF  
4599 4390

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/DataRespon  
4605 4396

src/main/activemq/wireformat/openwire/marshalsrv2/InboundTransactionWireMarshaller/openwire/marshal/v3/Destination  
4611 4402

src/main/activemq/wireformat/openwire/marshalsrv3/InboundTransactionWireMarshaller/openwire/marshal/v3/DiscoveryE  
4246 4408

src/main/activemq/wireformat/openwire/marshalsrv3/InboundTransactionWireMarshaller/openwire/marshal/v3/ExceptionR  
4252 4414

src/main/activemq/wireformat/openwire/marshaller/v3/FlushContinuationMarshaller/openwire/marshaller/v3/ShutDownIn	4420	4582
src/main/activemq/wireformat/openwire/marshaller/v3/InflightResponseMarshaller/openwire/marshaller/v3/Subscription	4426	4588
src/main/activemq/wireformat/openwire/marshaller/v3/JainActiveQueueAckMarshaller/openwire/marshaller/v3/Transaction	4432	4594
src/main/activemq/wireformat/openwire/marshaller/v3/JainActiveQueueAckMarshaller/openwire/marshaller/v3/Transaction	4438	4600
src/main/activemq/wireformat/openwire/marshaller/v3/JainActiveQueueMarshaller/openwire/marshaller/v3/WireFormat	4444	4606
src/main/activemq/wireformat/openwire/marshaller/v3/JainActiveQueueMarshaller/openwire/marshaller/v3/XATransact	4450	4612
src/main/activemq/wireformat/openwire/marshaller/v3/KeepActiveInfoMarshaller/openwire/marshaller/v4/ActiveMQB	4456	4247
src/main/activemq/wireformat/openwire/marshaller/v3/LastPartialContinuationMarshaller/openwire/marshaller/v4/ActiveMQB	4462	4253
src/main/activemq/wireformat/openwire/marshaller/v3/LastPartialContinuationMarshaller/openwire/marshaller/v4/ActiveMQD	4468	4259
src/main/activemq/wireformat/openwire/marshaller/v3/MarshallerFactory/openwire/marshaller/v4/ActiveMQM	4474	4265
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQM	4480	4271
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQC	4486	4277
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQC	4492	4283
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQS	4498	4289
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQT	4504	4295
src/main/activemq/wireformat/openwire/marshaller/v3/MisactiveQueueMarshaller/openwire/marshaller/v4/ActiveMQT	4510	4301
src/main/activemq/wireformat/openwire/marshaller/v3/NioBrokerBridgeFilterMarshaller/openwire/marshaller/v4/ActiveMQT	4516	4307
src/main/activemq/wireformat/openwire/marshaller/v3/PartialContinuationMarshaller/openwire/marshaller/v4/ActiveMQT	4522	4313
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerAckMarshaller/openwire/marshaller/v4/ActiveMQT	4528	4319
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/BaseComm	4534	4325
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/BrokerIdMa	4540	4331
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerInfoMarshaller/openwire/marshaller/v4/BrokerInfoM	4546	4337
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerSubscriptionInfoMarshaller/openwire/marshaller/v4/Connection	4552	4343
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerSubscriptionInfoMarshaller/openwire/marshaller/v4/Connection	4558	4349
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerSubscriptionInfoMarshaller/openwire/marshaller/v4/Connection	4564	4355
src/main/activemq/wireformat/openwire/marshaller/v3/SessionIdMarshaller/openwire/marshaller/v4/Connection	4570	4361
src/main/activemq/wireformat/openwire/marshaller/v3/SessionInfoMarshaller/openwire/marshaller/v4/ConsumerC	4576	4367

src/main/activemq/wireformat/openwire/marshaller/v4/ConsumerIdMarshaller.dat/openwire/marshaller/v4/ProducerIdMarshaller.dat 4373 4535

src/main/activemq/wireformat/openwire/marshaller/v4/ConsumerInfoMarshaller.dat/openwire/marshaller/v4/ProducerInfoMarshaller.dat 4379 4541

src/main/activemq/wireformat/openwire/marshaller/v4/ControlCommandMarshaller/openwire/marshaller/v4/RemoveInfoMarshaller 4385 4547

src/main/activemq/wireformat/openwire/marshaller/v4/DataActiveResponseMarshaller/openwire/marshaller/v4/RemoveSubscriptionMarshaller 4391 4553

src/main/activemq/wireformat/openwire/marshaller/v4/DataResponseMarshaller.dat/openwire/marshaller/v4/ReplayCommandMarshaller 4397 4559

src/main/activemq/wireformat/openwire/marshaller/v4/DeactivateInfoMarshaller/openwire/marshaller/v4/ResponseMarshaller 4403 4565

src/main/activemq/wireformat/openwire/marshaller/v4/DisconnectCommandMarshaller/openwire/marshaller/v4/SessionIdMarshaller 4409 4571

src/main/activemq/wireformat/openwire/marshaller/v4/ExceptionResponseMarshaller/openwire/marshaller/v4/SessionInfoMarshaller 4415 4577

src/main/activemq/wireformat/openwire/marshaller/v4/FlushActivemqWireFormat/openwire/marshaller/v4/ShutDownInfoMarshaller 4421 4583

src/main/activemq/wireformat/openwire/marshaller/v4/InformerResponseMarshaller/openwire/marshaller/v4/SubscriptionMarshaller 4427 4589

src/main/activemq/wireformat/openwire/marshaller/v4/JoinAndQueueAckMarshaller/openwire/marshaller/v4/TransactionMarshaller 4433 4595

src/main/activemq/wireformat/openwire/marshaller/v4/JoinAndTopicAckMarshaller/openwire/marshaller/v4/TransactionMarshaller 4439 4601

src/main/activemq/wireformat/openwire/marshaller/v4/JoinAndTransMarshaller.dat/openwire/marshaller/v4/WireFormatMarshaller 4445 4607

src/main/activemq/wireformat/openwire/marshaller/v4/JoinAndTransQueueMarshaller/openwire/marshaller/v4/XATransactionMarshaller 4451 4613

src/main/activemq/wireformat/openwire/marshaller/v4/KeepActiveInfoMarshaller.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4457 4248

src/main/activemq/wireformat/openwire/marshaller/v4/LastActiveQueueIdMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4463 4254

src/main/activemq/wireformat/openwire/marshaller/v4/LoadAndTransQueueMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4469 4260

src/main/activemq/wireformat/openwire/marshaller/v4/MarshallerFactory.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4475 4266

src/main/activemq/wireformat/openwire/marshaller/v4/MessageAckMarshaller.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4481 4272

src/main/activemq/wireformat/openwire/marshaller/v4/MessageDispatchMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4487 4278

src/main/activemq/wireformat/openwire/marshaller/v4/MessageDispatchNotificationMarshaller.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4493 4284

src/main/activemq/wireformat/openwire/marshaller/v4/MessageIdMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4499 4290

src/main/activemq/wireformat/openwire/marshaller/v4/MessageMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4505 4296

src/main/activemq/wireformat/openwire/marshaller/v4/MessagePushMarshaller.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4511 4302

src/main/activemq/wireformat/openwire/marshaller/v4/NewAckBridgeFilterMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4517 4308

src/main/activemq/wireformat/openwire/marshaller/v4/PartialCommandMarshaller/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4523 4314

src/main/activemq/wireformat/openwire/marshaller/v4/PublishAckMarshaller.dat/openwire/marshaller/v5/ActiveMQBrokerMarshaller 4529 4320

src/main/activemq/wireformat/openwire/marshalsrv5/BaseActiveAndMarshalHeader/openwire/marshal/v5/MessageDis	4326	4488
src/main/activemq/wireformat/openwire/marshalsrv5/BrokerIdMarshaller/openwire/marshal/v5/MessageDis	4332	4494
src/main/activemq/wireformat/openwire/marshalsrv5/BrokerInfoMarshaller/openwire/marshal/v5/MessageDis	4338	4500
src/main/activemq/wireformat/openwire/marshalsrv5/ConnectiveControlMarshaller/openwire/marshal/v5/MessageMa	4344	4506
src/main/activemq/wireformat/openwire/marshalsrv5/ConnectiveErrorMarshaller/openwire/marshal/v5/MessagePul	4350	4512
src/main/activemq/wireformat/openwire/marshalsrv5/ConnectiveIdMarshaller/openwire/marshal/v5/NetworkBri	4356	4518
src/main/activemq/wireformat/openwire/marshalsrv5/ConnectiveInfoMarshaller/openwire/marshal/v5/PartialCom	4362	4524
src/main/activemq/wireformat/openwire/marshalsrv5/ConsumerGroupWireFormat/openwire/marshal/v5/ProducerAc	4368	4530
src/main/activemq/wireformat/openwire/marshalsrv5/ConsumerIdMarshaller/openwire/marshal/v5/ProducerId	4374	4536
src/main/activemq/wireformat/openwire/marshalsrv5/ConsumerInfoMarshaller/openwire/marshal/v5/ProducerInf	4380	4542
src/main/activemq/wireformat/openwire/marshalsrv5/ControlCommandWireFormat/openwire/marshal/v5/RemoveInfo	4386	4548
src/main/activemq/wireformat/openwire/marshalsrv5/DataActiveResponseMarshaller/openwire/marshal/v5/RemoveSub	4392	4554
src/main/activemq/wireformat/openwire/marshalsrv5/DataResponseMarshaller/openwire/marshal/v5/ReplayCom	4398	4560
src/main/activemq/wireformat/openwire/marshalsrv5/DeactivatedInfoMarshaller/openwire/marshal/v5/ResponseM	4404	4566
src/main/activemq/wireformat/openwire/marshalsrv5/DiscoveryEventMarshaller/openwire/marshal/v5/SessionIdM	4410	4572
src/main/activemq/wireformat/openwire/marshalsrv5/ExceptionResponseMarshaller/openwire/marshal/v5/SessionInfo	4416	4578
src/main/activemq/wireformat/openwire/marshalsrv5/FlushActiveMQWireFormat/openwire/marshal/v5/ShutDownIn	4422	4584
src/main/activemq/wireformat/openwire/marshalsrv5/InActiveResponseMarshaller/openwire/marshal/v5/Subscription	4428	4590
src/main/activemq/wireformat/openwire/marshalsrv5/MainActiveQueueAckMarshaller/openwire/marshal/v5/Transaction	4434	4596
src/main/activemq/wireformat/openwire/marshalsrv5/MainActiveTopicAckMarshaller/openwire/marshal/v5/Transaction	4440	4602
src/main/activemq/wireformat/openwire/marshalsrv5/MainActiveTransMarshaller/openwire/marshal/v5/WireFormat	4446	4608
src/main/activemq/wireformat/openwire/marshalsrv5/MainActiveTransWireMarshaller/openwire/marshal/v5/XATransact	4452	4614
src/main/activemq/wireformat/openwire/marshalsrv5/MainActiveInfoMarshaller/openwire/marshal/v6/ActiveMQB	4458	4249
src/main/activemq/wireformat/openwire/marshalsrv5/MainPartialQueueWireFormat/openwire/marshal/v6/ActiveMQB	4464	4255
src/main/activemq/wireformat/openwire/marshalsrv5/MainTransWireIdMarshaller/openwire/marshal/v6/ActiveMQD	4470	4261
src/main/activemq/wireformat/openwire/marshalsrv5/MarshallerFactory/openwire/marshal/v6/ActiveMQM	4476	4267
src/main/activemq/wireformat/openwire/marshalsrv5/MessageAckMarshaller/openwire/marshal/v6/ActiveMQM	4482	4273

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQObjectMessageMarshaller/openwire/marshaller/v6/JournalTopic  
4279 4441

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQQueueMessageMarshaller/openwire/marshaller/v6/JournalTopic  
4285 4447

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQSimpleMessageMarshaller/openwire/marshaller/v6/JournalTopic  
4291 4453

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempDestinationMarshaller/openwire/marshaller/v6/KeepAliveIn  
4297 4459

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempQueueMarshaller/openwire/marshaller/v6/LastPartial  
4303 4465

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempTopicMarshaller/openwire/marshaller/v6/LocalTrans  
4309 4471

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTextMessageMarshaller/openwire/marshaller/v6/MarshallerF  
4315 4477

src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicMarshaller/openwire/marshaller/v6/MessageAck  
4321 4483

src/main/activemq/wireformat/openwire/marshaller/v6/BaseActiveMQMarshaller/openwire/marshaller/v6/MessageDis  
4327 4489

src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller/openwire/marshaller/v6/MessageDis  
4333 4495

src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller/openwire/marshaller/v6/MessageIdM  
4339 4501

src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionControlMarshaller/openwire/marshaller/v6/MessageMa  
4345 4507

src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionErrorMarshaller/openwire/marshaller/v6/MessagePul  
4351 4513

src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionIdMarshaller/openwire/marshaller/v6/NetworkBri  
4357 4519

src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionInfoMarshaller/openwire/marshaller/v6/PartialCom  
4363 4525

src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerGroupMarshaller/openwire/marshaller/v6/ProducerAc  
4369 4531

src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerIdMarshaller/openwire/marshaller/v6/ProducerId  
4375 4537

src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerInfoMarshaller/openwire/marshaller/v6/ProducerInf  
4381 4543

src/main/activemq/wireformat/openwire/marshaller/v6/DefaultCommandMarshaller/openwire/marshaller/v6/RemoveInfo  
4387 4549

src/main/activemq/wireformat/openwire/marshaller/v6/DataActiveResponseMarshaller/openwire/marshaller/v6/RemoveSub  
4393 4555

src/main/activemq/wireformat/openwire/marshaller/v6/DataResponseMarshaller/openwire/marshaller/v6/ReplayCom  
4399 4561

src/main/activemq/wireformat/openwire/marshaller/v6/DestinationInfoMarshaller/openwire/marshaller/v6/ResponseM  
4405 4567

src/main/activemq/wireformat/openwire/marshaller/v6/DisconnectErrorMarshaller/openwire/marshaller/v6/SessionIdM  
4411 4573

src/main/activemq/wireformat/openwire/marshaller/v6/ExceptionResponseMarshaller/openwire/marshaller/v6/SessionInfo  
4417 4579

src/main/activemq/wireformat/openwire/marshaller/v6/FlushCommandMarshaller/openwire/marshaller/v6/Shut downIn  
4423 4585

src/main/activemq/wireformat/openwire/marshaller/v6/LargeResponseMarshaller/openwire/marshaller/v6/Subscription  
4429 4591

src/main/activemq/wireformat/openwire/marshaller/v6/LocalQueueAckMarshaller/openwire/marshaller/v6/Transaction  
4435 4597



- src/main/activemq/wireformat/openwire/marshaller/InvalidDestinationException.h, 4603
- src/main/activemq/wireformat/openwire/marshaller/InvalidSelectorException.h, 4609
- src/main/activemq/wireformat/openwire/marshaller/MapMessage.h, 4615
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageConsumer.h, 4616
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageEnumeration.h, 4617
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageEOFException.h, 4618
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageFormatException.h, 4619
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageListener.h, 4620
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageNotReadableException.h, 4621
- src/main/activemq/wireformat/openwire/OpenWireFormat/MessageNotWriteableException.h, 4622
- src/main/activemq/wireformat/openwire/Utils/BooleanStream.h, 4623
- src/main/activemq/wireformat/openwire/Utils/HexTable.h, 4624
- src/main/activemq/wireformat/openwire/Utils/MessagePropertyInterceptor.h, 4625
- src/main/activemq/wireformat/stomp/StompCommandConstants.h, 4626
- src/main/activemq/wireformat/stomp/StompFrame.h, 4627
- src/main/activemq/wireformat/stomp/StompHelper.h, 4628
- src/main/activemq/wireformat/stomp/StompWireFormat.h, 4629
- src/main/activemq/wireformat/stomp/StompWireFormatFactory.h, 4630
- src/main/activemq/wireformat/WireFormat.h, 4631
- src/main/activemq/wireformat/WireFormatFactory.h, 4632
- src/main/activemq/wireformat/WireFormatNegotiator.h, 4633
- src/main/activemq/wireformat/WireFormatRegistry.h, 4634
- src/main/cms/BytesMessage.h, 4635
- src/main/cms/Closeable.h, 4636
- src/main/cms/CMSException.h, 4637
- src/main/cms/CMSPProperties.h, 4638
- src/main/cms/CMSSecurityException.h, 4639
- src/main/cms/Config.h, 4640
- src/main/cms/Connection.h, 4641
- src/main/cms/ConnectionFactory.h, 4642
- src/main/cms/ConnectionMetaData.h, 4643
- src/main/cms/DeliveryMode.h, 4644
- src/main/cms/Destination.h, 4645
- src/main/cms/ExceptionListener.h, 4646
- src/main/cms/IllegalStateException.h, 4647
- src/main/cms/InvalidClientIdException.h, 4648
- src/main/cms/InvalidSelectorException.h, 4649
- src/main/cms/MapMessage.h, 4650
- src/main/cms/MessageConsumer.h, 4651
- src/main/cms/MessageEnumeration.h, 4652
- src/main/cms/MessageEOFException.h, 4653
- src/main/cms/MessageFormatException.h, 4654
- src/main/cms/MessageListener.h, 4655
- src/main/cms/MessageNotReadableException.h, 4656
- src/main/cms/MessageNotWriteableException.h, 4657
- src/main/cms/MessageProducer.h, 4658
- src/main/cms/ObjectMessage.h, 4659
- src/main/cms/Queue.h, 4660
- src/main/cms/QueueBrowser.h, 4661
- src/main/cms/Session.h, 4662
- src/main/cms/Startable.h, 4663
- src/main/cms/Stopable.h, 4664
- src/main/cms/StreamMessage.h, 4665
- src/main/cms/TemporaryQueue.h, 4666
- src/main/cms/TemporaryTopic.h, 4667
- src/main/cms/TextMessage.h, 4668
- src/main/cms/Topic.h, 4669
- src/main/cms/UnsupportedOperationException.h, 4670
- src/main/decaf/internal/AprPool.h, 4671
- src/main/decaf/internal/DecafRuntime.h, 4672
- src/main/decaf/internal/io/StandardErrorOutputStream.h, 4673
- src/main/decaf/internal/io/StandardInputStream.h, 4674
- src/main/decaf/internal/io/StandardOutputStream.h, 4675
- src/main/decaf/internal/net/DefaultServerSocketFactory.h, 4676
- src/main/decaf/internal/net/DefaultSocketFactory.h, 4677
- src/main/decaf/internal/net/Network.h, 4678
- src/main/decaf/internal/net/SocketFileDescriptor.h, 4679
- src/main/decaf/internal/net/ssl/DefaultSSLContext.h, 4680
- src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h, 4681
- src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h, 4682
- src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h, 4683
- src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h, 4684

src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h, 4686  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h, 4687  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h, 4688  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h, 4689  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h, 4690  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h, 4691  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h, 4692  
 src/main/decaf/internal/net/tcp/TcpSocket.h, 4693  
 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h, 4694  
 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h, 4695  
 src/main/decaf/internal/net/URIEncoderDecoder.h, 4696  
 src/main/decaf/internal/net/URIHelper.h, 4697  
 src/main/decaf/internal/net/URIType.h, 4698  
 src/main/decaf/internal/nio/BufferFactory.h, 4699  
 src/main/decaf/internal/nio/ByteBuffer.h, 4700  
 src/main/decaf/internal/nio/CharArrayBuffer.h, 4701  
 src/main/decaf/internal/nio/DoubleArrayBuffer.h, 4702  
 src/main/decaf/internal/nio/FloatArrayBuffer.h, 4703  
 src/main/decaf/internal/nio/IntArrayBuffer.h, 4704  
 src/main/decaf/internal/nio/LongArrayBuffer.h, 4705  
 src/main/decaf/internal/nio/ShortArrayBuffer.h, 4706  
 src/main/decaf/internal/security/unix/SecureRandomImpl.h, 4707  
 src/main/decaf/internal/security/windows/SecureRandomImpl.h, 4708  
 src/main/decaf/internal/util/ByteArrayAdapter.h, 4709  
 src/main/decaf/internal/util/concurrent/ConditionImpl.h, 4710  
 src/main/decaf/internal/util/concurrent/MutexImpl.h, 4711  
 src/main/decaf/internal/util/concurrent/SynchronizationImpl.h, 4712  
 src/main/decaf/internal/util/concurrent/TransferQueue.h, 4713  
 src/main/decaf/internal/util/concurrent/TransferQueue.h, 4714  
 src/main/decaf/internal/util/concurrent/TransferStack.h, 4715  
 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h, 4716  
 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h, 4717  
 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h, 4718  
 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h, 4719  
 src/main/decaf/internal/util/GenericResource.h, 4720  
 src/main/decaf/internal/util/HexStringParser.h, 4721  
 src/main/decaf/internal/util/Resource.h, 4722  
 src/main/decaf/internal/util/ResourceLifecycleManager.h, 4723  
 src/main/decaf/internal/util/TimerTaskHeap.h, 4724  
 src/main/decaf/internal/util/zip/crc32.h, 4725  
 src/main/decaf/internal/util/zip/deflate.h, 4726  
 src/main/decaf/internal/util/zip/gzguts.h, 4727  
 src/main/decaf/internal/util/zip/infast.h, 4728  
 src/main/decaf/internal/util/zip/infix.h, 4729  
 src/main/decaf/internal/util/zip/inflate.h, 4730  
 src/main/decaf/internal/util/zip/inftrees.h, 4731  
 src/main/decaf/internal/util/zip/trees.h, 4732  
 src/main/decaf/internal/util/zip/zconf.h, 4733  
 src/main/decaf/internal/util/zip/zlib.h, 4734  
 src/main/decaf/internal/util/zip/zutil.h, 4735  
 src/main/decaf/io/BlockingByteArrayInputStream.h, 4736  
 src/main/decaf/io/BufferedInputStream.h, 4737  
 src/main/decaf/io/BufferedOutputStream.h, 4738  
 src/main/decaf/io/ByteArrayInputStream.h, 4739  
 src/main/decaf/io/ByteArrayOutputStream.h, 4740  
 src/main/decaf/io/Closeable.h, 4741  
 src/main/decaf/io/DataInput.h, 4742  
 src/main/decaf/io/DataInputStream.h, 4743  
 src/main/decaf/io/DataOutput.h, 4744  
 src/main/decaf/io/DataOutputStream.h, 4745

- src/main/decaf/io/EOFException.h, 4755
- src/main/decaf/io/FileDescriptor.h, 4756
- src/main/decaf/io/FilterInputStream.h, 4757
- src/main/decaf/io/FilterOutputStream.h, 4758
- src/main/decaf/io/Flushable.h, 4759
- src/main/decaf/io/InputStream.h, 4760
- src/main/decaf/io/InputStreamReader.h, 4761
- src/main/decaf/io/InterruptedIOException.h, 4762
- src/main/decaf/io/IOException.h, 4763
- src/main/decaf/io/OutputStream.h, 4764
- src/main/decaf/io/OutputStreamWriter.h, 4765
- src/main/decaf/io/PushbackInputStream.h, 4766
- src/main/decaf/io/Reader.h, 4767
- src/main/decaf/io/UnsupportedEncodingException.h, 4768
- src/main/decaf/io/UTFDataFormatException.h, 4769
- src/main/decaf/io/Writer.h, 4770
- src/main/decaf/lang/Appendable.h, 4771
- src/main/decaf/lang/ArrayPointer.h, 4772
- src/main/decaf/lang/Boolean.h, 4774
- src/main/decaf/lang/Byte.h, 4775
- src/main/decaf/lang/Character.h, 4776
- src/main/decaf/lang/CharSequence.h, 4777
- src/main/decaf/lang/Comparable.h, 4778
- src/main/decaf/lang/Double.h, 4779
- src/main/decaf/lang/Exception.h, 4780
- src/main/decaf/lang/exceptions/ClassCastException.h, 4781
- src/main/decaf/lang/exceptions/ExceptionDefines.h, 4173
- src/main/decaf/lang/exceptions/IllegalArgumentException.h, 4782
- src/main/decaf/lang/exceptions/IllegalMonitorStateException.h, 4783
- src/main/decaf/lang/exceptions/IllegalStateException.h, 4645
- src/main/decaf/lang/exceptions/IllegalThreadStateException.h, 4784
- src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h, 4785
- src/main/decaf/lang/exceptions/InterruptedException.h, 4786
- src/main/decaf/lang/exceptions/InvalidStateException.h, 4787
- src/main/decaf/lang/exceptions/NoSuchElementException.h, 4788
- src/main/decaf/lang/exceptions/NullPointerException.h, 4789
- src/main/decaf/lang/exceptions/NumberFormatException.h, 4790
- src/main/decaf/lang/exceptions/RuntimeException.h, 4791
- src/main/decaf/lang/exceptions/UnsupportedOperationException.h, 4671
- src/main/decaf/lang/Float.h, 4792
- src/main/decaf/lang/Integer.h, 4793
- src/main/decaf/lang/Iterable.h, 4794
- src/main/decaf/lang/Long.h, 4795
- src/main/decaf/lang/Math.h, 4796
- src/main/decaf/lang/Number.h, 4797
- src/main/decaf/lang/Pointer.h, 4798
- src/main/decaf/lang/Readable.h, 4799
- src/main/decaf/lang/Runnable.h, 4800
- src/main/decaf/lang/Runtime.h, 4801
- src/main/decaf/lang/Short.h, 4802
- src/main/decaf/lang/String.h, 4803
- src/main/decaf/lang/System.h, 4804
- src/main/decaf/lang/Thread.h, 4805
- src/main/decaf/lang/ThreadGroup.h, 4806
- src/main/decaf/lang/Throwable.h, 4807
- src/main/decaf/net/BindException.h, 4808
- src/main/decaf/net/ConnectException.h, 4809
- src/main/decaf/net/HttpRetryException.h, 4810
- src/main/decaf/net/Inet4Address.h, 4811
- src/main/decaf/net/Inet6Address.h, 4812
- src/main/decaf/net/InetAddress.h, 4813
- src/main/decaf/net/InetSocketAddress.h, 4814
- src/main/decaf/net/MalformedURLException.h, 4815
- src/main/decaf/net/NoRouteToHostException.h, 4816
- src/main/decaf/net/PortUnreachableException.h, 4817
- src/main/decaf/net/ProtocolException.h, 4818
- src/main/decaf/net/ServerSocket.h, 4819
- src/main/decaf/net/ServerSocketFactory.h, 4820
- src/main/decaf/net/Socket.h, 4821
- src/main/decaf/net/SocketAddress.h, 4822
- src/main/decaf/net/SocketError.h, 4823
- src/main/decaf/net/SocketException.h, 4824
- src/main/decaf/net/SocketFactory.h, 4825
- src/main/decaf/net/SocketImpl.h, 4826
- src/main/decaf/net/SocketImplFactory.h, 4827
- src/main/decaf/net/SocketOptions.h, 4828
- src/main/decaf/net/SocketTimeoutException.h, 4829
- src/main/decaf/net/ssl/SSLContext.h, 4830
- src/main/decaf/net/ssl/SSLContextSpi.h, 4831
- src/main/decaf/net/ssl/SSLParameters.h, 4832
- src/main/decaf/net/ssl/SSLServerSocket.h, 4833

- src/main/decaf/net/ssl/SSLServerSocketFactory.h, 4834
- src/main/decaf/net/ssl/SSLSocket.h, 4835
- src/main/decaf/net/ssl/SSLSocketFactory.h, 4836
- src/main/decaf/net/UnknownHostException.h, 4837
- src/main/decaf/net/UnknownServiceException.h, 4838
- src/main/decaf/net/URI.h, 4839
- src/main/decaf/net/URISyntaxException.h, 4840
- src/main/decaf/net/URL.h, 4841
- src/main/decaf/net/URLDecoder.h, 4842
- src/main/decaf/net/URLEncoder.h, 4843
- src/main/decaf/nio/Buffer.h, 4844
- src/main/decaf/nio/BufferOverflowException.h, 4845
- src/main/decaf/nio/BufferUnderflowException.h, 4846
- src/main/decaf/nio/ByteBuffer.h, 4847
- src/main/decaf/nio/CharBuffer.h, 4848
- src/main/decaf/nio/DoubleBuffer.h, 4849
- src/main/decaf/nio/FloatBuffer.h, 4850
- src/main/decaf/nio/IntBuffer.h, 4851
- src/main/decaf/nio/InvalidMarkException.h, 4852
- src/main/decaf/nio/LongBuffer.h, 4853
- src/main/decaf/nio/ReadOnlyBufferException.h, 4854
- src/main/decaf/nio/ShortBuffer.h, 4855
- src/main/decaf/security/auth/x500/X500Principal.h, 4856
- src/main/decaf/security/cert/Certificate.h, 4857
- src/main/decaf/security/cert/CertificateEncodingException.h, 4858
- src/main/decaf/security/cert/CertificateException.h, 4859
- src/main/decaf/security/cert/CertificateExpiredException.h, 4860
- src/main/decaf/security/cert/CertificateNotYetValidException.h, 4861
- src/main/decaf/security/cert/CertificateParsingException.h, 4862
- src/main/decaf/security/cert/X509Certificate.h, 4863
- src/main/decaf/security/GeneralSecurityException.h, 4864
- src/main/decaf/security/InvalidKeyException.h, 4865
- src/main/decaf/security/Key.h, 4866
- src/main/decaf/security/KeyException.h, 4867
- src/main/decaf/security/KeyManagementException.h, 4868
- src/main/decaf/security/NoSuchAlgorithmException.h, 4869
- src/main/decaf/security/NoSuchProviderException.h, 4870
- src/main/decaf/security/Principal.h, 4871
- src/main/decaf/security/PublicKey.h, 4872
- src/main/decaf/security/SecureRandom.h, 4873
- src/main/decaf/security/SecureRandomSpi.h, 4874
- src/main/decaf/security/SignatureException.h, 4875
- src/main/decaf/util/AbstractCollection.h, 4876
- src/main/decaf/util/AbstractList.h, 4877
- src/main/decaf/util/AbstractMap.h, 4878
- src/main/decaf/util/AbstractQueue.h, 4879
- src/main/decaf/util/AbstractSequentialList.h, 4880
- src/main/decaf/util/AbstractSet.h, 4881
- src/main/decaf/util/Collection.h, 4882
- src/main/decaf/util/Comparator.h, 4883
- src/main/decaf/util/comparators/Less.h, 4884
- src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 4885
- src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 4886
- src/main/decaf/util/concurrent/atomic/AtomicReferenceCounter.h, 4887
- src/main/decaf/util/concurrent/atomic/AtomicReference.h, 4888
- src/main/decaf/util/concurrent/BlockingQueue.h, 4889
- src/main/decaf/util/concurrent/BrokenBarrierException.h, 4890
- src/main/decaf/util/concurrent/Callable.h, 4891
- src/main/decaf/util/concurrent/CancellationException.h, 4892
- src/main/decaf/util/concurrent/Concurrent.h, 4893
- src/main/decaf/util/concurrent/ConcurrentMap.h, 4894
- src/main/decaf/util/concurrent/ConcurrentStlMap.h, 4895
- src/main/decaf/util/concurrent/CountDownLatch.h, 4896
- src/main/decaf/util/concurrent/Delayed.h, 4897
- src/main/decaf/util/concurrent/ExecutionException.h, 4898
- src/main/decaf/util/concurrent/Executor.h, 4899

- src/main/decaf/util/concurrent/ExecutorService.h, 4900
- src/main/decaf/util/concurrent/Future.h, 4901
- src/main/decaf/util/concurrent/Lock.h, 4902
- src/main/decaf/util/concurrent/locks/Condition.h, 4904
- src/main/decaf/util/concurrent/locks/Lock.h, 4903
- src/main/decaf/util/concurrent/locks/LockSupport.h, 4905
- src/main/decaf/util/concurrent/locks/ReadWriteLock.h, 4906
- src/main/decaf/util/concurrent/locks/ReentrantLock.h, 4907
- src/main/decaf/util/concurrent/Mutex.h, 4908
- src/main/decaf/util/concurrent/PooledThread.h, 4909
- src/main/decaf/util/concurrent/PooledThreadListener.h, 4910
- src/main/decaf/util/concurrent/RejectedExecutionException.h, 4911
- src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 4912
- src/main/decaf/util/concurrent/Semaphore.h, 4913
- src/main/decaf/util/concurrent/Synchronizable.h, 4914
- src/main/decaf/util/concurrent/SynchronousQueue.h, 4915
- src/main/decaf/util/concurrent/TaskListener.h, 4916
- src/main/decaf/util/concurrent/ThreadFactory.h, 4917
- src/main/decaf/util/concurrent/ThreadPool.h, 4918
- src/main/decaf/util/concurrent/TimeoutException.h, 4919
- src/main/decaf/util/concurrent/TimeUnit.h, 4920
- src/main/decaf/util/Config.h, 4229
- src/main/decaf/util/Date.h, 4921
- src/main/decaf/util/Iterator.h, 4922
- src/main/decaf/util/List.h, 4923
- src/main/decaf/util/ListIterator.h, 4924
- src/main/decaf/util/logging/ConsoleHandler.h, 4925
- src/main/decaf/util/logging/EventManager.h, 4926
- src/main/decaf/util/logging/Filter.h, 4927
- src/main/decaf/util/logging/Formatter.h, 4928
- src/main/decaf/util/logging/Handler.h, 4929
- src/main/decaf/util/logging/Level.h, 4930
- src/main/decaf/util/logging/Logger.h, 4931
- src/main/decaf/util/logging/LoggerCommon.h, 4932
- src/main/decaf/util/logging/LoggerDefines.h, 4933
- src/main/decaf/util/logging/LoggerHierarchy.h, 4935
- src/main/decaf/util/logging/LogManager.h, 4936
- src/main/decaf/util/logging/LogRecord.h, 4937
- src/main/decaf/util/logging/LogWriter.h, 4938
- src/main/decaf/util/logging/MarkBlockLogger.h, 4939
- src/main/decaf/util/logging/PropertiesChangeListener.h, 4940
- src/main/decaf/util/logging/SimpleFormatter.h, 4941
- src/main/decaf/util/logging/SimpleLogger.h, 4942
- src/main/decaf/util/logging/StreamHandler.h, 4943
- src/main/decaf/util/logging/XMLFormatter.h, 4944
- src/main/decaf/util/Map.h, 4945
- src/main/decaf/util/PriorityQueue.h, 4946
- src/main/decaf/util/Properties.h, 4947
- src/main/decaf/util/Queue.h, 4660
- src/main/decaf/util/Random.h, 4948
- src/main/decaf/util/Set.h, 4949
- src/main/decaf/util/StlList.h, 4950
- src/main/decaf/util/StlMap.h, 4951
- src/main/decaf/util/StlQueue.h, 4952
- src/main/decaf/util/StlSet.h, 4953
- src/main/decaf/util/StringTokenizer.h, 4954
- src/main/decaf/util/Timer.h, 4955
- src/main/decaf/util/TimerTask.h, 4956
- src/main/decaf/util/UUID.h, 4957
- src/main/decaf/util/zip/Adler32.h, 4958
- src/main/decaf/util/zip/CheckedInputStream.h, 4959
- src/main/decaf/util/zip/CheckedOutputStream.h, 4960
- src/main/decaf/util/zip/Checksum.h, 4961
- src/main/decaf/util/zip/CRC32.h, 4962
- src/main/decaf/util/zip/DataFormatException.h, 4963
- src/main/decaf/util/zip/Deflater.h, 4964
- src/main/decaf/util/zip/DeflaterOutputStream.h, 4965
- src/main/decaf/util/zip/Inflater.h, 4966
- src/main/decaf/util/zip/InflaterInputStream.h, 4967
- src/main/decaf/util/zip/ZipException.h, 4968
- SSLContext

- decaf::net::ssl::SSLContext, 3545
- SSLParameters
  - decaf::net::ssl::SSLParameters, 3551, 3552
- SSLServerSocket
  - decaf::net::ssl::SSLServerSocket, 3555, 3556
- SSLServerSocketFactory
  - decaf::net::ssl::SSLServerSocketFactory, 3561
- SSLSocket
  - decaf::net::ssl::SSLSocket, 3564, 3565
- SSLSocketFactory
  - decaf::net::ssl::SSLSocketFactory, 3572
- SslTransport
  - activemq::transport::tcp::SslTransport, 3574
- stackTrace
  - decaf::lang::Exception, 1837
- StandardErrorOutputStream
  - decaf::internal::io::StandardErrorOutputStream, 3580
- StandardInputStream
  - decaf::internal::io::StandardInputStream, 3582
- StandardOutputStream
  - decaf::internal::io::StandardOutputStream, 3584
- start
  - activemq::commands::ConsumerControl, 1405
  - activemq::core::ActiveMQConnection, 290
  - activemq::core::ActiveMQConsumer, 318
  - activemq::core::ActiveMQSession, 531
  - activemq::core::ActiveMQSessionExecutor, 535
  - activemq::core::MessageDispatchChannel, 2603
  - activemq::transport::correlator::ResponseCorrelator, 3294
  - activemq::transport::failover::FailoverTransport, 1884
  - activemq::transport::IOTransport, 2150
  - activemq::transport::mock::MockTransport, 2777
  - activemq::transport::Transport, 3888
  - activemq::transport::TransportFilter, 3897
  - activemq::wireformat::openwire::OpenWireFormat, 2904
  - cms::Startable, 3586
  - decaf::lang::Thread, 3772
  - gz\_state, 1977
- startHandshake
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2869
  - decaf::net::ssl::SSLSocket, 3570
- stat\_desc
  - tree\_desc\_s, 3905
- State
  - decaf::lang::Thread, 3768
- state
  - z\_stream\_s, 4062
- static\_dtree
  - trees.h, 4736
- static\_len
  - internal\_state, 2120
- static\_ltree
  - trees.h, 4736
- static\_tree\_desc
  - deflate.h, 4727
- STATIC\_TREES
  - zutil.h, 4745
- staticCast
  - decaf::lang::Pointer, 2952
- StaticInitializer
  - activemq::core::ActiveMQConstants::StaticInitializer, 3588
- status
  - internal\_state, 2120
- std, 178
- std::binary\_function, 835
- std::less< decaf::lang::ArrayPointer< T > >, 2330
  - operator(), 2330
- std::less< decaf::lang::Pointer< T > >, 2331
  - operator(), 2331
- StlList
  - decaf::util::StlList, 3593, 3594
- StlMap
  - decaf::util::StlMap, 3605, 3606
- StlQueue
  - decaf::util::StlQueue, 3617
- StlSet
  - decaf::util::StlSet, 3625
- StompFrame
  - activemq::wireformat::stomp::StompFrame, 3634
- StompHelper
  - activemq::wireformat::stomp::StompHelper, 3639
- StompWireFormat
  - activemq::wireformat::stomp::StompWireFormat, 3644
- StompWireFormatFactory
  - activemq::wireformat::stomp::StompWireFormatFactory, 3647
- stop
  - activemq::commands::ConsumerControl, 1405
  - activemq::core::ActiveMQConnection, 290

- activemq::core::ActiveMQConsumer, 318
- activemq::core::ActiveMQSession, 531
- activemq::core::ActiveMQSessionExecutor, 535
- activemq::core::MessageDispatchChannel, 2603
- activemq::transport::failover::FailoverTransport, 1884
- activemq::transport::IOTransport, 2151
- activemq::transport::mock::MockTransport, 2777
- activemq::transport::Transport, 3888
- activemq::transport::TransportFilter, 3898
- cms::Stoppable, 3648
- decaf::util::concurrent::PooledThread, 2969
- store
  - decaf::util::Properties, 3132, 3133
- STORED
  - inflate.h, 4733
- STORED\_BLOCK
  - zutil.h, 4745
- strategy
  - gz\_state, 1977
  - internal\_state, 2120
- StreamHandler
  - decaf::util::logging::StreamHandler, 3650
- String
  - decaf::lang::String, 3666
- STRING\_TYPE
  - activemq::util::PrimitiveValueNode, 3016
- StringTokenizer
  - decaf::util::StringTokenizer, 3668
- stringValue
  - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 3009
- strm
  - gz\_state, 1977
  - internal\_state, 2120
- strstart
  - internal\_state, 2120
- subscriptionName
  - activemq::commands::RemoveSubscriptionInfo, 3226
  - activemq::commands::SubscriptionInfo, 3675
- SUBSCRIBE
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- subscribedDestination
  - activemq::commands::SubscriptionInfo, 3675
- SubscriptionInfo
  - activemq::commands::SubscriptionInfo, 3672
- SubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SubscriptionInfo, 3681
  - activemq::wireformat::openwire::marshal::v2::SubscriptionInfo, 3697
  - activemq::wireformat::openwire::marshal::v3::SubscriptionInfo, 3677
  - activemq::wireformat::openwire::marshal::v4::SubscriptionInfo, 3689
  - activemq::wireformat::openwire::marshal::v5::SubscriptionInfo, 3685
  - activemq::wireformat::openwire::marshal::v6::SubscriptionInfo, 3693
- subscriptionName
  - activemq::commands::ConsumerInfo, 1466
- subscriptionName
  - activemq::commands::JournalTopicAck, 2187
- subSequence
  - decaf::internal::nio::CharArrayBuffer, 1117
  - decaf::lang::CharSequence, 1136
  - decaf::lang::String, 3667
  - decaf::nio::CharBuffer, 1132
- supportsUrgentData
  - decaf::net::SocketImpl, 3536
- suspend
  - activemq::commands::ConnectionControl, 1271
- swap
  - decaf::lang::ArrayPointer, 736
  - decaf::lang::Pointer, 2952
  - decaf::util::concurrent::atomic::AtomicRefCounter, 748
- sync
  - inflate.h, 4733
- sync
  - decaf::io::FileDescriptor, 1892
- SynchronizableImpl
  - decaf::internal::util::concurrent::SynchronizableImpl, 3712
- synchronized
  - Concurrent.h, 4893
- SynchronousQueue
  - decaf::util::concurrent::SynchronousQueue, 3718
- syncRequest
  - activemq::core::ActiveMQConnection, 290
  - activemq::core::ActiveMQSession, 531
- System
  - decaf::lang::System, 3728
- TABLE
  - inflate.h, 4733
- take

- decaf::util::concurrent::BlockingQueue, 848
- decaf::util::concurrent::SynchronousQueue, 3724
- takeSession
  - activemq::cmsutil::SessionPool, 3436
- targetConsumerId
  - activemq::commands::Message, 2532
- Task
  - decaf::util::concurrent::ThreadPool, 3779
- TcpSocket
  - decaf::internal::net::tcp::TcpSocket, 3741
- TcpSocketInputStream
  - decaf::internal::net::tcp::TcpSocketInputStream, 3748
- TcpSocketOutputStream
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3750
- TcpTransport
  - activemq::transport::tcp::TcpTransport, 3753
- TEMP\_POSTFIX
  - activemq::commands::ActiveMQDestination, 332
- TEMP\_PREFIX
  - activemq::commands::ActiveMQDestination, 332
- TEMP\_QUEUE\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 332
- TEMP\_TOPIC\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 332
- TEMPORARY\_QUEUE
  - cms::Destination, 1723
- TEMPORARY\_TOPIC
  - cms::Destination, 1723
- TEMPQUEUE\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- TEMPTOPIC\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- TERMINATED
  - decaf::lang::Thread, 3768
- TEXT
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- text
  - activemq::commands::ActiveMQTextMessage, 666
  - gz\_header\_s, 1975
- Thread
  - decaf::lang::Thread, 3768
- ThreadGroup
  - decaf::lang::ThreadGroup, 3776
- ThreadPool
  - decaf::util::concurrent::ThreadPool, 3779
- Throwable
  - decaf::lang::Throwable, 3784
- Throwing
  - decaf::util::logging, 176
- throwing
  - decaf::util::logging::Logger, 2396
- tightMarshal
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 819
  - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1638
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller, 215
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller, 255
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 339
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller, 380
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 407
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller, 452
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 497
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 558
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller, 586
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller, 615
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller, 648
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller, 677
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 705
  - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 782
  - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 882
  - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 914
  - activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller, 1282
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1314
  - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1345
  - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1375





activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller; openwire::marshal::v2::MessageDispatcher	803	2641
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller; openwire::marshal::v2::MessageIdMarshaller	894	2670
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller; openwire::marshal::v2::MessageMarshaller	926	2705
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller; openwire::marshal::v2::MessagePullMarshaller	1294	2746
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller; openwire::marshal::v2::NetworkBridgeFilter	1302	2798
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller; openwire::marshal::v2::PartialCommandMarshaller	1333	2928
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller; openwire::marshal::v2::ProducerAckMarshaller	1363	3042
activemq::wireformat::openwire::marshal::v2::ConsumerGroupViewMarshaller; openwire::marshal::v2::ProducerIdMarshaller	1408	3074
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller; openwire::marshal::v2::ProducerInfoMarshaller	1437	3107
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller; openwire::marshal::v2::RemoveInfoMarshaller	1470	3200
activemq::wireformat::openwire::marshal::v2::ContactGroupAndIdMarshaller; openwire::marshal::v2::RemoveSubscriptionMarshaller	1499	3237
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller; openwire::marshal::v2::ReplayCommandMarshaller	1533	3264
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller; openwire::marshal::v2::ResponseMarshaller	1596	3302
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller; openwire::marshal::v2::SessionIdMarshaller	1734	3385
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller; openwire::marshal::v2::SessionInfoMarshaller	1768	3429
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller; openwire::marshal::v2::ShutdownInfoMarshaller	1848	3479
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller; openwire::marshal::v2::SubscriptionInfoMarshaller	1946	3698
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller; openwire::marshal::v2::TransactionIdMarshaller	2100	3833
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller; openwire::marshal::v2::TransactionInfoMarshaller	2165	3873
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller; openwire::marshal::v2::WireFormatInfoMarshaller	2194	4002
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller; openwire::marshal::v2::XATransactionInfoMarshaller	2217	4042
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller; openwire::marshal::v3::ActiveMQBlobMarshaller	2248	211
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller; openwire::marshal::v3::ActiveMQBytesMarshaller	2275	251
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller; openwire::marshal::v3::ActiveMQDestinationMarshaller	2314	335
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller; openwire::marshal::v3::ActiveMQMapMarshaller	2357	376
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller; openwire::marshal::v3::ActiveMQMessageMarshaller	2571	403
activemq::wireformat::openwire::marshal::v2::MessageDispatcherMarshaller; openwire::marshal::v3::ActiveMQObjectMarshaller	2608	448

activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	493	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	2221
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	554	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	2252
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueDeflationMarshaller	582	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	2279
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueFormatMarshaller	611	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	2310
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueTopicMarshaller	640	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	2361
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	669	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2575
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	697	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2612
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	768	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2645
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	874	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2682
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	906	activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2700
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1274	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2754
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1306	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureMarshaller	2810
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1337	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2936
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1367	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	3050
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	1412	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	3082
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1441	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	3119
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1474	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	3208
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	1503	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	3233
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	1537	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	3268
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1600	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	3312
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1738	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	3401
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller	1772	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	3425
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1852	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	3491
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1950	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3678
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller	2104	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3837
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	2173	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3861
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	2198	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	4014

activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1742
4054	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireMessageMarshaller	1776
219	
activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryWireMessageMarshaller	1860
259	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1954
343	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	2108
384	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	2177
411	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireMessageMarshaller	2206
456	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireMarshaller	2229
501	
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireMessageMarshaller	2260
562	
activemq::wireformat::openwire::marshal::v4::ActiveMQSimpleInformationMarshaller	2283
590	
activemq::wireformat::openwire::marshal::v4::ActiveMQSimpleQueueMarshaller	2322
619	
activemq::wireformat::openwire::marshal::v4::ActiveMQSimpleTopicMarshaller	2369
644	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2579
673	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2620
701	
activemq::wireformat::openwire::marshal::v4::BaseCommandWireMarshaller	2649
775	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2674
878	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2710
910	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2758
1278	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2814
1310	
activemq::wireformat::openwire::marshal::v4::ConnectionMarshaller	2940
1341	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	3046
1371	
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	3078
1416	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	3103
1445	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	3220
1478	
activemq::wireformat::openwire::marshal::v4::ContractCommandMarshaller	3249
1507	
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	3256
1541	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	3297
1604	

activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1379	activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	1379
3389		activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	1424
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1424	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	1453
3433		activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	1486
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1453	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	1515
3495		activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1549
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1486	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	1588
3690		activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1754
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1515	activemq::wireformat::openwire::marshal::v5::DiscoveryEventManager	1784
3841		activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1856
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1549	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1962
3869		activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller	2116
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1588	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	2169
4006		activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	2190
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1754	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	2237
4046		activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	2256
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	1784	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	2287
227		activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	2318
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	1856	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller	2365
263		activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2587
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1962	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2616
347		activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2657
activemq::wireformat::openwire::marshal::v5::ActiveMQMapWireFormatMarshaller	2116	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2678
388		activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2695
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2169	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2750
415		activemq::wireformat::openwire::marshal::v5::NetworkBridgeFactory	2806
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireFormatMarshaller	2190	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2932
460			
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueWireFormatMarshaller	2237		
505			
activemq::wireformat::openwire::marshal::v5::ActiveMQStackingMessageMarshaller	2256		
566			
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	2287		
594			
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2318		
623			
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2365		
652			
activemq::wireformat::openwire::marshal::v5::ActiveMQTextWireFormatMarshaller	2587		
681			
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2616		
709			
activemq::wireformat::openwire::marshal::v5::BaseCommandWireFormatMarshaller	2657		
789			
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2678		
886			
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2695		
918			
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2750		
1286			
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2806		
1318			
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	2932		
1349			

activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	713
3054	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	796
3086	
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	890
3115	
activemq::wireformat::openwire::marshal::v5::RemoteInfoMarshaller	922
3216	
activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshaller	1290
3245	
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller	1322
3276	
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	1353
3307	
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	1383
3397	
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	1428
3417	
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	1457
3487	
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	1490
3686	
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	1519
3825	
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	1553
3853	
activemq::wireformat::openwire::marshal::v5::WireFormatMarshaller	1592
3994	
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	1750
4058	
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	1764
231	
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	1844
267	
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	1942
355	
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	2096
396	
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	2161
423	
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	2202
468	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	2225
513	
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceMessageMarshaller	2244
574	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	2271
602	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	2306
631	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	2353
660	
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	2567
685	

activemq::wireformat::openwire::marshal::v6::MessagePatchMarshaller,  
     2628  
 activemq::wireformat::openwire::marshal::v6::MessagePatchNotificationMarshaller,  
     2637  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller,  
     2686  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller,  
     2720  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller,  
     2766  
 activemq::wireformat::openwire::marshal::v6::NetworkFilterMarshaller,  
     2802  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller,  
     2924  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller,  
     3058  
 activemq::wireformat::openwire::marshal::v6::ProducerMarshaller,  
     3090  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,  
     3123  
 activemq::wireformat::openwire::marshal::v6::RemoveMarshaller,  
     3204  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller,  
     3241  
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller,  
     3272  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,  
     3322  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,  
     3393  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,  
     3413  
 activemq::wireformat::openwire::marshal::v6::ShutdownMarshaller,  
     3475  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,  
     3694  
 activemq::wireformat::openwire::marshal::v6::TransactionMarshaller,  
     3845  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,  
     3865  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller,  
     3998  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,  
     4038  
 tightMarshal2  
     1545  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,  
     819  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller,  
     1645  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBlockWireFormatMarshaller,  
     215  
 activemq::wireformat::openwire::marshal::v1::ActiveMQByteMessageMarshaller,  
     255  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,  
     1864  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller,  
     1339  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,  
     1380  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,  
     1407  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller,  
     1552  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,  
     1977  
 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller,  
     1558  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,  
     1586  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTemporalityMarshaller,  
     1613  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTemporalityMarshaller,  
     1648  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller,  
     1677  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,  
     1705  
 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller,  
     1733  
 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller,  
     1882  
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller,  
     1914  
 activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller,  
     1982  
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller,  
     2014  
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller,  
     2145  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller,  
     2175  
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller,  
     2420  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller,  
     2449  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller,  
     2482  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller,  
     2510  
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller,  
     2555  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller,  
     2608  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller,  
     2746  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller,  
     2780  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller,  
     2864  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller,  
     2915

- 1958
- activemq::wireformat::openwire::marshal::v1::IntegerResponseInfoMarshaler 3682
- 2112
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaler 3829
- 2181
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckInfoMarshaler 3857
- 2210
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckInfoMarshaler 4010
- 2233
- activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler 4050
- 2264
- activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler 223
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaler 271
- 2291
- activemq::wireformat::openwire::marshal::v1::LastReceivedCommandInfoMarshaler 351
- 2326
- activemq::wireformat::openwire::marshal::v1::LocalTransactionInfoMarshaler 392
- 2373
- activemq::wireformat::openwire::marshal::v1::MessageAckInfoMarshaler 419
- 2583
- activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaler 464
- 2624
- activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaler 509
- 2653
- activemq::wireformat::openwire::marshal::v1::MessageIdInfoMarshaler 570
- 2690
- activemq::wireformat::openwire::marshal::v1::MessageIdInfoMarshaler 598
- 2715
- activemq::wireformat::openwire::marshal::v1::MessageIdInfoMarshaler 627
- 2762
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaler 656
- 2818
- activemq::wireformat::openwire::marshal::v1::PartialCommandInfoMarshaler 689
- 2944
- activemq::wireformat::openwire::marshal::v1::ProducerIdInfoMarshaler 717
- 3062
- activemq::wireformat::openwire::marshal::v1::ProducerIdInfoMarshaler 804
- 3094
- activemq::wireformat::openwire::marshal::v1::ProducerIdInfoMarshaler 894
- 3111
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaler 926
- 3212
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaler 1294
- 3229
- activemq::wireformat::openwire::marshal::v1::ReplyCommandInfoMarshaler 1302
- 3260
- activemq::wireformat::openwire::marshal::v1::ResponseInfoMarshaler 1333
- 3318
- activemq::wireformat::openwire::marshal::v1::SessionIdInfoMarshaler 1363
- 3405
- activemq::wireformat::openwire::marshal::v1::SessionIdInfoMarshaler 1408
- 3421
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaler 1437
- 3483
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaler



1470	3200
activemq::wireformat::openwire::marshal::v2::ContainerCommandMarshaller	openwire::marshal::v2::RemoveSubscription
1499	3237
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	openwire::marshal::v2::ReplayCommand
1533	3264
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	openwire::marshal::v2::ResponseMarshaller
1596	3303
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	openwire::marshal::v2::SessionIdMarshaller
1734	3385
activemq::wireformat::openwire::marshal::v2::DiscoveryInfoMarshaller	openwire::marshal::v2::SessionInfoMarshaller
1768	3429
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	openwire::marshal::v2::ShutdownInfoMarshaller
1848	3479
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	openwire::marshal::v2::SubscriptionInfoMarshaller
1946	3698
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	openwire::marshal::v2::TransactionIdMarshaller
2100	3833
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	openwire::marshal::v2::TransactionInfoMarshaller
2165	3873
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	openwire::marshal::v2::WireFormatInfoMarshaller
2194	4002
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	openwire::marshal::v2::XATransactionInfoMarshaller
2217	4042
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	openwire::marshal::v3::ActiveMQBlobMarshaller
2248	211
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	openwire::marshal::v3::ActiveMQBytesMarshaller
2275	251
activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller	openwire::marshal::v3::ActiveMQDestinationMarshaller
2314	335
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	openwire::marshal::v3::ActiveMQMapMarshaller
2357	376
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	openwire::marshal::v3::ActiveMQMessageMarshaller
2571	403
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller	openwire::marshal::v3::ActiveMQObjectMarshaller
2608	448
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller	openwire::marshal::v3::ActiveMQQueueMarshaller
2641	493
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	openwire::marshal::v3::ActiveMQStreamMarshaller
2670	554
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	openwire::marshal::v3::ActiveMQTemplateMarshaller
2705	582
activemq::wireformat::openwire::marshal::v2::MessagePublishInfoMarshaller	openwire::marshal::v3::ActiveMQTemplateMarshaller
2746	611
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	openwire::marshal::v3::ActiveMQTemplateMarshaller
2798	640
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	openwire::marshal::v3::ActiveMQTextMarshaller
2928	669
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	openwire::marshal::v3::ActiveMQTopicMarshaller
3042	697
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	openwire::marshal::v3::BaseCommandMarshaller
3074	769
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	openwire::marshal::v3::BrokerIdMarshaller
3107	874
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	openwire::marshal::v3::BrokerInfoMarshaller

906	2700
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
1274	2754
activemq::wireformat::openwire::marshal::v3::ConnectionErrorInfoMarshaller	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureInfoMarshaller
1306	2810
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1337	2936
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1367	3050
activemq::wireformat::openwire::marshal::v3::ConsumerGroupViewMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1412	3082
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1441	3119
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1474	3208
activemq::wireformat::openwire::marshal::v3::ContainerCommandMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1503	3233
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1537	3268
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1600	3313
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1738	3401
activemq::wireformat::openwire::marshal::v3::DisconnectInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1772	3425
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1852	3491
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1950	3678
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller
2104	3837
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
2173	3861
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
2198	4014
activemq::wireformat::openwire::marshal::v3::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller
2221	4054
activemq::wireformat::openwire::marshal::v3::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
2252	219
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
2279	259
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
2310	343
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller
2361	384
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
2575	411
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
2612	456
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
2645	501
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller
2682	562
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTemplateMarshaller

590	2283
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaler	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaler
619	2322
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaler	activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaler
644	2369
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaler	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaler
673	2579
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaler	activemq::wireformat::openwire::marshal::v4::MessageDispatchInfoMarshaler
701	2620
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaler	activemq::wireformat::openwire::marshal::v4::MessageDispatchInfoMarshaler
776	2649
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaler	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaler
878	2674
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaler	activemq::wireformat::openwire::marshal::v4::MessageMarshaler
910	2710
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaler	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaler
1278	2758
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaler	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFailureMarshaler
1310	2814
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaler	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaler
1341	2940
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaler	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaler
1371	3046
activemq::wireformat::openwire::marshal::v4::ConsumerGroupInfoMarshaler	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaler
1416	3078
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaler	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaler
1445	3103
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaler	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaler
1478	3220
activemq::wireformat::openwire::marshal::v4::ContactCommandMarshaler	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaler
1507	3249
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaler	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaler
1541	3256
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaler	activemq::wireformat::openwire::marshal::v4::ResponseMarshaler
1604	3298
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaler	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaler
1742	3389
activemq::wireformat::openwire::marshal::v4::DiscardInfoMarshaler	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaler
1776	3433
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaler	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaler
1860	3495
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaler	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaler
1954	3690
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaler	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaler
2108	3841
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaler	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaler
2177	3869
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaler	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaler
2206	4006
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaler	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaler
2229	4046
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaler	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMarshaler
2260	227
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaler	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMarshaler

263  
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller; openwire::marshal::v5::FlushCommandMarshaller  
 347  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller; openwire::marshal::v5::IntegerResponseMarshaller  
 388  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller; openwire::marshal::v5::JournalQueueAckMarshaller  
 415  
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller; openwire::marshal::v5::JournalTopicAckMarshaller  
 460  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller; openwire::marshal::v5::JournalTraceMarshaller  
 505  
 activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller; openwire::marshal::v5::JournalTransactionMarshaller  
 566  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller; openwire::marshal::v5::KeepAliveInfoMarshaller  
 594  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller; openwire::marshal::v5::LastPartialCommandMarshaller  
 623  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller; openwire::marshal::v5::LocalTransactionMarshaller  
 652  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller; openwire::marshal::v5::MessageAckMarshaller  
 681  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller; openwire::marshal::v5::MessageDispatchMarshaller  
 709  
 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller; openwire::marshal::v5::MessageDispatchMarshaller  
 790  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller; openwire::marshal::v5::MessageIdMarshaller  
 886  
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller; openwire::marshal::v5::MessageMarshaller  
 918  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller; openwire::marshal::v5::MessagePullMarshaller  
 1286  
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller; openwire::marshal::v5::NetworkBridgeMarshaller  
 1318  
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller; openwire::marshal::v5::PartialCommandMarshaller  
 1349  
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller; openwire::marshal::v5::ProducerAckMarshaller  
 1379  
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller; openwire::marshal::v5::ProducerIdMarshaller  
 1424  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller; openwire::marshal::v5::ProducerInfoMarshaller  
 1453  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller; openwire::marshal::v5::RemoveInfoMarshaller  
 1486  
 activemq::wireformat::openwire::marshal::v5::ContractCommandMarshaller; openwire::marshal::v5::RemoveSubscriptionMarshaller  
 1515  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller; openwire::marshal::v5::ReplayCommandMarshaller  
 1549  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller; openwire::marshal::v5::ResponseMarshaller  
 1588  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller; openwire::marshal::v5::SessionIdMarshaller  
 1754  
 activemq::wireformat::openwire::marshal::v5::DiscardRequestMarshaller; openwire::marshal::v5::SessionInfoMarshaller  
 1784  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller; openwire::marshal::v5::ShutdownInfoMarshaller

3487	1457
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	openwire::marshal::v6::ConsumerInfoMarshaller
3686	1490
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	openwire::marshal::v6::ControlCommandMarshaller
3825	1519
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	openwire::marshal::v6::DataArrayResponseMarshaller
3853	1553
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	openwire::marshal::v6::DataResponseMarshaller
3994	1592
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	openwire::marshal::v6::DestinationInfoMarshaller
4058	1750
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobWireFormatMarshaller	openwire::marshal::v6::DiscoveryEventManager
231	1764
activemq::wireformat::openwire::marshal::v6::ActiveMQByteWireFormatMarshaller	openwire::marshal::v6::ExceptionResponseMarshaller
267	1844
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	openwire::marshal::v6::FlushCommandMarshaller
355	1942
activemq::wireformat::openwire::marshal::v6::ActiveMQMapWireFormatMarshaller	openwire::marshal::v6::IntegerResponseMarshaller
396	2096
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	openwire::marshal::v6::JournalQueueAckMarshaller
423	2161
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectWireFormatMarshaller	openwire::marshal::v6::JournalTopicAckMarshaller
468	2202
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueWireFormatMarshaller	openwire::marshal::v6::JournalTraceMarshaller
513	2225
activemq::wireformat::openwire::marshal::v6::ActiveMQSequenceWireFormatMarshaller	openwire::marshal::v6::JournalTransactionMarshaller
574	2244
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	openwire::marshal::v6::KeepAliveInfoMarshaller
602	2271
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	openwire::marshal::v6::LastPartialCommandMarshaller
631	2306
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	openwire::marshal::v6::LocalTransactionMarshaller
660	2353
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	openwire::marshal::v6::MessageAckMarshaller
685	2567
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	openwire::marshal::v6::MessageDispatchMarshaller
713	2628
activemq::wireformat::openwire::marshal::v6::BaseCommandWireFormatMarshaller	openwire::marshal::v6::MessageDispatchMarshaller
797	2637
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	openwire::marshal::v6::MessageIdMarshaller
890	2686
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller	openwire::marshal::v6::MessageMarshaller
922	2720
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	openwire::marshal::v6::MessagePullMarshaller
1290	2766
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	openwire::marshal::v6::NetworkBridgeMarshaller
1322	2802
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	openwire::marshal::v6::PartialCommandMarshaller
1353	2924
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	openwire::marshal::v6::ProducerAckMarshaller
1383	3058
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	openwire::marshal::v6::ProducerIdMarshaller
1428	3090
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	openwire::marshal::v6::ProducerInfoMarshaller

3123	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller,	823
3204	tightMarshalObjectArray2
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller,	823
3241	activemq::wireformat::openwire::marshal::v6::ReplyToGroupAndIdMarshaller,
3272	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,	824
3323	tightMarshalString2
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,	824
3393	activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,
3413	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::ShutdownMarshaller,	824
3475	activemq::wireformat::openwire::marshal::DataStreamMarshaller,
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,	824
3694	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller,
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,	824
3845	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller,
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,	825
3865	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller,	840
3998	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller,
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,	881
4038	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,
tightMarshalBrokerError1	408
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	408
820	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller,
tightMarshalBrokerError2	453
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	453
820	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,
tightMarshalCachedObject1	498
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	498
820	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller,
tightMarshalCachedObject2	559
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	559
820	activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,
tightMarshalCachedObject2	587
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	587
821	activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,
tightMarshalLong1	649
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	649
821	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller,
tightMarshalLong2	678
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	678
822	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,
tightMarshalNestedObject1	706
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	706
822	activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller,
activemq::wireformat::openwire::OpenWireFormat,	784
2897	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller,
tightMarshalNestedObject2	883
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	883
822	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller,
activemq::wireformat::openwire::OpenWireFormat,	915
2897	activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller,
tightMarshalObjectArray1	1288
	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller,
	1315
	activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller,
	1346

activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	3063
1376	
activemq::wireformat::openwire::marshal::v1::ConsumerGroupViewMarshaller	3095
1421	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	3112
1450	
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	3213
1483	
activemq::wireformat::openwire::marshal::v1::ContainerCommandMarshaller	3230
1512	
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	3261
1546	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	3318
1609	
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	3406
1747	
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	3422
1781	
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3484
1865	
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3683
1959	
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	3830
2113	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3858
2182	
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	4011
2211	
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	4051
2234	
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	224
2265	
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	272
2292	
activemq::wireformat::openwire::marshal::v1::LastBatchCompInfoMarshaller	352
2327	
activemq::wireformat::openwire::marshal::v1::LocalTransactionMarshaller	393
2374	
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	420
2584	
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	465
2625	
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	510
2654	
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	571
2691	
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	599
2716	
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	628
2763	
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	657
2819	
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	690
2945	

activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	2609	activemq::wireformat::openwire::marshal::v2::MessageDispatcher
718		
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2642	activemq::wireformat::openwire::marshal::v2::MessageDispatcher
805		
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2671	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller
895		
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2706	activemq::wireformat::openwire::marshal::v2::MessageMarshaller
927		
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2747	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller
1295		
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2799	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilter
1303		
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2929	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller
1334		
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	3043	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller
1364		
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	3075	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller
1409		
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	3108	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller
1438		
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3201	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller
1471		
activemq::wireformat::openwire::marshal::v2::ContactGroupMarshaller	3238	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller
1500		
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	3265	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
1534		
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3303	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller
1597		
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	3386	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller
1735		
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	3430	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller
1769		
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3480	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller
1849		
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3699	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller
1947		
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3834	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller
2101		
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3874	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller
2166		
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	4003	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller
2195		
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	4043	activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller
2218		
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	212	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller
2249		
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	252	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller
2276		
activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller	336	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
2315		
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	377	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller
2358		
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	404	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
2572		



activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	2199	activemq::wireformat::openwire::marshal::v3::JournalTopicAck
449		
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	2222	activemq::wireformat::openwire::marshal::v3::JournalTraceMa
494		
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceMessageMarshaller	2253	activemq::wireformat::openwire::marshal::v3::JournalTransact
555		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDeflationMarshaller	2280	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoM
583		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	2311	activemq::wireformat::openwire::marshal::v3::LastPartialComr
612		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2362	activemq::wireformat::openwire::marshal::v3::LocalTransaction
641		
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2576	activemq::wireformat::openwire::marshal::v3::MessageAckMar
670		
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2613	activemq::wireformat::openwire::marshal::v3::MessageDispat ch
698		
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	2646	activemq::wireformat::openwire::marshal::v3::MessageDispat ch
770		
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2683	activemq::wireformat::openwire::marshal::v3::MessageIdMarsh
875		
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2701	activemq::wireformat::openwire::marshal::v3::MessageMarshal
907		
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2755	activemq::wireformat::openwire::marshal::v3::MessagePullMar
1275		
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2811	activemq::wireformat::openwire::marshal::v3::NetworkBridgeF
1307		
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2937	activemq::wireformat::openwire::marshal::v3::PartialCommand
1338		
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	3051	activemq::wireformat::openwire::marshal::v3::ProducerAckMar
1368		
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	3083	activemq::wireformat::openwire::marshal::v3::ProducerIdMars
1413		
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	3120	activemq::wireformat::openwire::marshal::v3::ProducerInfoMa
1442		
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	3209	activemq::wireformat::openwire::marshal::v3::RemoveInfoMars
1475		
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	3234	activemq::wireformat::openwire::marshal::v3::RemoveSubscrip
1504		
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	3269	activemq::wireformat::openwire::marshal::v3::ReplayCommand
1538		
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	3313	activemq::wireformat::openwire::marshal::v3::ResponseMarshal
1601		
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	3402	activemq::wireformat::openwire::marshal::v3::SessionIdMarsha
1739		
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	3426	activemq::wireformat::openwire::marshal::v3::SessionInfoMars
1773		
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	3492	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMa
1853		
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3679	activemq::wireformat::openwire::marshal::v3::SubscriptionInfo
1951		
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller	3838	activemq::wireformat::openwire::marshal::v3::TransactionIdMa
2105		
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	3862	activemq::wireformat::openwire::marshal::v3::TransactionInfo
2174		

activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	openwire::marshal::v4::DataResponseM
4015	1605
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	openwire::marshal::v4::DestinationInfoM
4055	1743
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMessageMarshaller	openwire::marshal::v4::DiscoveryEventM
220	1777
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMessageMarshaller	openwire::marshal::v4::ExceptionRespon
260	1861
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationWireFormatMessageMarshaller	openwire::marshal::v4::FlushCommandL
344	1955
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMessageMarshaller	openwire::marshal::v4::IntegerResponse
385	2109
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	openwire::marshal::v4::JournalQueueAc
412	2178
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMessageMarshaller	openwire::marshal::v4::JournalTopicAck
457	2207
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMessageMarshaller	openwire::marshal::v4::JournalTraceMa
502	2230
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMessageMarshaller	openwire::marshal::v4::JournalTransact
563	2261
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	openwire::marshal::v4::KeepAliveInfoM
591	2284
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	openwire::marshal::v4::LastPartialComm
620	2323
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	openwire::marshal::v4::LocalTransaction
645	2370
activemq::wireformat::openwire::marshal::v4::ActiveMQTextWireFormatMessageMarshaller	openwire::marshal::v4::MessageAckMar
674	2580
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	openwire::marshal::v4::MessageDispatc
702	2621
activemq::wireformat::openwire::marshal::v4::BaseCommandWireFormatMessageMarshaller	openwire::marshal::v4::MessageDispatc
777	2650
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	openwire::marshal::v4::MessageIdMarsh
879	2675
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	openwire::marshal::v4::MessageMarshal
911	2711
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	openwire::marshal::v4::MessagePullMar
1279	2759
activemq::wireformat::openwire::marshal::v4::ConnectionFromMarshaller	openwire::marshal::v4::NetworkBridgeF
1311	2815
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	openwire::marshal::v4::PartialCommand
1342	2941
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	openwire::marshal::v4::ProducerAckMar
1372	3047
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	openwire::marshal::v4::ProducerIdMars
1417	3079
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	openwire::marshal::v4::ProducerInfoMa
1446	3104
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	openwire::marshal::v4::RemoveInfoMars
1479	3221
activemq::wireformat::openwire::marshal::v4::ContractCommandMarshaller	openwire::marshal::v4::RemoveSubscrip
1508	3250
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	openwire::marshal::v4::ReplayCommand
1542	3257

activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1350
3298	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1380
3390	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1425
3434	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1454
3496	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1487
3691	
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1516
3842	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1550
3870	
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller	1589
4007	
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1755
4047	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	1785
228	
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	1857
264	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1963
348	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	2117
389	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	2170
416	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	2191
461	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	2238
506	
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	2257
567	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	2288
595	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2319
624	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2366
653	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2588
682	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2617
710	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2658
791	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2679
887	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2696
919	
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2751
1287	
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2807
1319	

activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	openwire::marshal::v6::ActiveMQTextM
2933	686
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	openwire::marshal::v6::ActiveMQTopicC
3055	714
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	openwire::marshal::v6::BaseCommandM
3087	798
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	openwire::marshal::v6::BrokerIdMarsha
3116	891
activemq::wireformat::openwire::marshal::v5::RemoteInfoMarshaller	openwire::marshal::v6::BrokerInfoMarsh
3217	923
activemq::wireformat::openwire::marshal::v5::RemoteSubscriptionInfoMarshaller	openwire::marshal::v6::ConnectionCont
3246	1291
activemq::wireformat::openwire::marshal::v5::ReplaceCommandMarshaller	openwire::marshal::v6::ConnectionError
3277	1323
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	openwire::marshal::v6::ConnectionIdMa
3308	1354
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	openwire::marshal::v6::ConnectionInfoM
3398	1384
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	openwire::marshal::v6::ConsumerContro
3418	1429
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	openwire::marshal::v6::ConsumerIdMar
3488	1458
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	openwire::marshal::v6::ConsumerInfoMa
3687	1491
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	openwire::marshal::v6::ControlComman
3826	1520
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	openwire::marshal::v6::DataArrayRespo
3854	1554
activemq::wireformat::openwire::marshal::v5::WireFormatMarshaller	openwire::marshal::v6::DataResponseM
3995	1593
activemq::wireformat::openwire::marshal::v5::XATransactionMarshaller	openwire::marshal::v6::DestinationInfoM
4059	1751
activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller	openwire::marshal::v6::DiscoveryEventM
232	1765
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	openwire::marshal::v6::ExceptionRespor
268	1845
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	openwire::marshal::v6::FlushCommandL
356	1943
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	openwire::marshal::v6::IntegerResponse
397	2097
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	openwire::marshal::v6::JournalQueueAc
424	2162
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	openwire::marshal::v6::JournalTopicAck
469	2203
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	openwire::marshal::v6::JournalTraceMa
514	2226
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	openwire::marshal::v6::JournalTransact
575	2245
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	openwire::marshal::v6::KeepAliveInfoM
603	2272
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	openwire::marshal::v6::LastPartialComr
632	2307
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	openwire::marshal::v6::LocalTransaction
661	2354

- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 826
- 2568 tightUnmarshalConstByteArray
- activemq::wireformat::openwire::marshal::v6::MessageDispatcherMarshaller, 826
- 2629 BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::v6::MessageDispatcherNotificationMarshaller, 826
- 2638 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 826
- 2687 tightUnmarshalNestedObject
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 827
- 2721 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2897
- 2767 activemq::wireformat::openwire::OpenWireFormat,
- activemq::wireformat::openwire::marshal::v6::NioUnmarshalStringMarshaller, 2803
- 2803 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2925
- 2925 TIME
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3059
- 3059 time
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3091
- 3091 TIMED\_WAITING
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3124
- 3124 timedJoin
- activemq::wireformat::openwire::marshal::v6::RemoteInfoMarshaller, 3205
- 3205 timedWait
- activemq::wireformat::openwire::marshal::v6::RemoteSubscriptionInfoMarshaller, 3242
- 3242 timeout
- activemq::wireformat::openwire::marshal::v6::ReplyToQueueInfoMarshaller, 3273
- 3273 DestinationInfo,
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3323
- 3323 TimeoutException
- activemq::wireformat::openwire::marshal::v6::SessionMarshaller, 3394
- 3394 TimeoutException,
- 3787, 3788
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3414
- 3414 decaf::util::Timer, 3792
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3476
- 3476 TimerImpl
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3695
- 3695 TimerTask
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3846
- 3846 TimerTaskHeap
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3866
- 3866 timerTaskHeap
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3999
- 3999 decaf::util::UUID, 3973
- activemq::wireformat::openwire::marshal::v6::XmppActionIdMarshaller, 4039
- 4039 decaf::util::concurrent::TimeUnit, 3809
- tightUnmarshalBrokerError toArray
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 825
- 825 cms::CMSProperties, 1168
- tightUnmarshalByteArray decaf::util::AbstractCollection, 189
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 825
- 825 decaf::util::concurrent::SynchronousQueue,
- 3724
- tightUnmarshalCachedObject
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 3134
- 3134

- decaf::util::StlQueue, 3620
- decaf::util::StringTokenizer, 3670
- toBinaryString
  - decaf::lang::Integer, 2087
  - decaf::lang::Long, 2430
- toByteArray
  - decaf::io::ByteArrayOutputStream, 1029
- toDays
  - decaf::util::concurrent::TimeUnit, 3812
- toDegrees
  - decaf::lang::Math, 2511
- toDestinationOption
  - activemq::core::ActiveMQConstants, 309
- toHexFromBytes
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 828
- toHexString
  - decaf::lang::Double, 1796
  - decaf::lang::Float, 1912
  - decaf::lang::Integer, 2088
  - decaf::lang::Long, 2430
- toHours
  - decaf::util::concurrent::TimeUnit, 3812
- toMicros
  - decaf::util::concurrent::TimeUnit, 3813
- toMillis
  - decaf::util::concurrent::TimeUnit, 3813
- toMinutes
  - decaf::util::concurrent::TimeUnit, 3813
- toNanos
  - decaf::util::concurrent::TimeUnit, 3814
- toOctalString
  - decaf::lang::Integer, 2088
  - decaf::lang::Long, 2431
- TOPIC
  - cms::Destination, 1723
- TOPIC\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3631
- TOPIC\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 332
- toRadians
  - decaf::lang::Math, 2511
- toSeconds
  - decaf::util::concurrent::TimeUnit, 3814
- toStream
  - activemq::wireformat::stomp::StompFrame, 3637
- toString
  - activemq::commands::ActiveMQBlobMessage, 207
  - activemq::commands::ActiveMQBytesMessage, 244
- activemq::commands::ActiveMQDestination, 330
- activemq::commands::ActiveMQMapMessage, 372
- activemq::commands::ActiveMQMessage, 399
- activemq::commands::ActiveMQObjectMessage, 445
- activemq::commands::ActiveMQQueue, 486
- activemq::commands::ActiveMQStreamMessage, 547
- activemq::commands::ActiveMQTempDestination, 578
- activemq::commands::ActiveMQTempQueue, 607
- activemq::commands::ActiveMQTempTopic, 636
- activemq::commands::ActiveMQTextMessage, 666
- activemq::commands::ActiveMQTopic, 694
- activemq::commands::BaseCommand, 763
- activemq::commands::BaseDataStructure, 833
- activemq::commands::BooleanExpression, 856
- activemq::commands::BrokerId, 871
- activemq::commands::BrokerInfo, 901
- activemq::commands::Command, 1197
- activemq::commands::ConnectionControl, 1270
- activemq::commands::ConnectionError, 1298
- activemq::commands::ConnectionId, 1329
- activemq::commands::ConnectionInfo, 1359
- activemq::commands::ConsumerControl, 1404
- activemq::commands::ConsumerId, 1433
- activemq::commands::ConsumerInfo, 1464
- activemq::commands::ControlCommand, 1495
- activemq::commands::DataArrayResponse, 1530
- activemq::commands::DataResponse, 1585
- activemq::commands::DataStructure, 1663
- activemq::commands::DestinationInfo, 1730
- activemq::commands::DiscoveryEvent, 1760
- activemq::commands::ExceptionResponse, 1841
- activemq::commands::FlushCommand, 1939

- activemq::commands::IntegerResponse, 2093
- activemq::commands::JournalQueueAck, 2158
- activemq::commands::JournalTopicAck, 2186
- activemq::commands::JournalTrace, 2214
- activemq::commands::JournalTransaction, 2241
- activemq::commands::KeepAliveInfo, 2268
- activemq::commands::LastPartialCommand, 2302
- activemq::commands::LocalTransactionId, 2350
- activemq::commands::Message, 2530
- activemq::commands::MessageAck, 2563
- activemq::commands::MessageDispatch, 2597
- activemq::commands::MessageDispatchNotification, 2633
- activemq::commands::MessageId, 2666
- activemq::commands::MessagePull, 2742
- activemq::commands::NetworkBridgeFilter, 2795
- activemq::commands::PartialCommand, 2920
- activemq::commands::ProducerAck, 3038
- activemq::commands::ProducerId, 3070
- activemq::commands::ProducerInfo, 3099
- activemq::commands::RemoveInfo, 3196
- activemq::commands::RemoveSubscriptionInfo, 3225
- activemq::commands::ReplayCommand, 3253
- activemq::commands::Response, 3287
- activemq::commands::SessionId, 3382
- activemq::commands::SessionInfo, 3409
- activemq::commands::ShutdownInfo, 3472
- activemq::commands::SubscriptionInfo, 3674
- activemq::commands::TransactionId, 3821
- activemq::commands::TransactionInfo, 3849
- activemq::commands::WireFormatInfo, 3991
- activemq::commands::XATransactionId, 4034
- activemq::core::ActiveMQConstants, 309
- activemq::state::ConnectionState, 1391
- activemq::state::ConsumerState, 1492
- activemq::state::ProducerState, 3125
- activemq::state::SessionState, 3438
- activemq::state::TransactionState, 3876
- activemq::util::ActiveMQProperties, 481
- activemq::util::PrimitiveList, 2990
- activemq::util::PrimitiveMap, 3000
- activemq::util::PrimitiveValueNode, 3024
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 828
- cms::CMSProperties, 1168
- decaf::io::ByteArrayOutputStream, 1030
- decaf::io::FilterOutputStream, 1902
- decaf::io::InputStream, 2050
- decaf::io::OutputStream, 2911
- decaf::lang::Boolean, 853, 854
- decaf::lang::Byte, 966
- decaf::lang::Character, 1106
- decaf::lang::CharSequence, 1136
- decaf::lang::Double, 1797
- decaf::lang::Float, 1912, 1913
- decaf::lang::Integer, 2088, 2089
- decaf::lang::Long, 2431
- decaf::lang::Short, 3446
- decaf::lang::String, 3667
- decaf::lang::Thread, 3773
- decaf::net::InetAddress, 2022
- decaf::net::ServerSocket, 3360
- decaf::net::Socket, 3518
- decaf::net::SocketImpl, 3536
- decaf::net::URI, 3931
- decaf::nio::ByteBuffer, 1054
- decaf::nio::CharBuffer, 1133
- decaf::nio::DoubleBuffer, 1818
- decaf::nio::FloatBuffer, 1934
- decaf::nio::IntBuffer, 2075
- decaf::nio::LongBuffer, 2453
- decaf::nio::ShortBuffer, 3468
- decaf::security::cert::Certificate, 1088
- decaf::util::concurrent::atomic::AtomicBoolean, 739
- decaf::util::concurrent::atomic::AtomicInteger, 745
- decaf::util::concurrent::atomic::AtomicReference, 751
- decaf::util::concurrent::locks::ReentrantLock, 3186
- decaf::util::concurrent::Semaphore, 3347
- decaf::util::concurrent::TimeUnit, 3814
- decaf::util::Date, 1668
- decaf::util::logging::Level, 2335
- decaf::util::Properties, 3134
- decaf::util::UUID, 3974
- total
  - inflate\_state, 2025
- total\_in
  - z\_stream\_s, 4062
- total\_out
  - z\_stream\_s, 4062

- toURI
  - activemq::util::CompositeData, 1220
- toURIOption
  - activemq::core::ActiveMQConstants, 309
- toURL
  - decaf::net::URI, 3931
- Trace
  - zutil.h, 4745
- Tracec
  - zutil.h, 4745
- Tracecv
  - zutil.h, 4745
- Tracev
  - zutil.h, 4745
- Tracevv
  - zutil.h, 4745
- track
  - activemq::state::ConnectionStateTracker, 1398
- trackBack
  - activemq::state::ConnectionStateTracker, 1398
- Tracked
  - activemq::state::Tracked, 3818
- TRANSACTION\_STATE\_BEGIN
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_-COMMITONEPHASE
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_-COMMITTWOPHASE
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_END
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_FORGET
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_PREPARE
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_RECOVER
  - activemq::core::ActiveMQConstants, 309
- TRANSACTION\_STATE\_ROLLBACK
  - activemq::core::ActiveMQConstants, 309
- TransactionId
  - activemq::commands::TransactionId, 3820
- transactionId
  - activemq::commands::JournalTopicAck, 2187
  - activemq::commands::JournalTransaction, 2241
  - activemq::commands::Message, 2532
  - activemq::commands::MessageAck, 2564
  - activemq::commands::TransactionInfo, 3850
- TransactionIdMarshaller

- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3828
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3832
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3836
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3840
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3824
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3844
- TransactionInfo
  - activemq::commands::TransactionInfo, 3848
- TransactionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3856
  - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3872
  - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3860
  - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3868
  - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3852
  - activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3864
- TransactionState
  - activemq::core::ActiveMQConstants, 308
  - activemq::state::TransactionState, 3876
- transfer
  - decaf::internal::util::concurrent::TransferQueue, 3879
  - decaf::internal::util::concurrent::TransferStack, 3881, 3882
- TransferQueue
  - decaf::internal::util::concurrent::TransferQueue, 3878
- TransferStack
  - decaf::internal::util::concurrent::TransferStack, 3881
- TransportFilter
  - activemq::transport::TransportFilter, 3893
- transportInterrupted
  - activemq::core::ActiveMQConnection, 291
  - activemq::state::ConnectionStateTracker, 1398
  - activemq::transport::DefaultTransportListener, 1705
  - activemq::transport::failover::FailoverTransportListener, 1889
  - activemq::transport::TransportFilter, 3898



- activemq::transport::TransportListener, 3901
- transportResumed
  - activemq::core::ActiveMQConnection, 291
  - activemq::transport::DefaultTransportListener, 1705
  - activemq::transport::failover::FailoverTransportListener, 1889
  - activemq::transport::TransportFilter, 3898
  - activemq::transport::TransportListener, 3901
- tree\_desc
  - deflate.h, 4727
- tree\_desc\_s, 3905
  - dyn\_tree, 3905
  - max\_code, 3905
  - stat\_desc, 3905
- trees.h
  - \_dist\_code, 4735
  - \_length\_code, 4735
  - base\_dist, 4736
  - base\_length, 4736
  - static\_dtree, 4736
  - static\_ltree, 4736
- TRY\_FREE
  - zutil.h, 4745
- tryAcquire
  - decaf::util::concurrent::Semaphore, 3347–3349
- tryLock
  - activemq::core::MessageDispatchChannel, 2603
  - decaf::internal::util::concurrent::SynchronizableImpl, 3713
  - decaf::io::InputStream, 2051
  - decaf::io::OutputStream, 2911
  - decaf::util::AbstractCollection, 189
  - decaf::util::concurrent::ConcurrentStlMap, 1246
  - decaf::util::concurrent::locks::Lock, 2380, 2381
  - decaf::util::concurrent::locks::ReentrantLock, 3186, 3187
  - decaf::util::concurrent::Mutex, 2784
  - decaf::util::concurrent::Synchronizable, 3704
  - decaf::util::StlMap, 3612
  - decaf::util::StlQueue, 3620
- trylock
  - decaf::internal::util::concurrent::MutexImpl, 2789
- TYPE
  - inflate.h, 4732
- type
  - activemq::transport::TransportListener, 3901
  - activemq::commands::JournalTransaction, 2241
  - activemq::commands::Message, 2532
  - activemq::commands::TransactionInfo, 3850
  - TYPEDO
    - inflate.h, 4732
  - uch
    - zutil.h, 4745
  - uchf
    - zutil.h, 4745
  - uInt
    - zconf.h, 4738
  - uIntf
    - zconf.h, 4738
  - ulg
    - zutil.h, 4745
  - uLong
    - zconf.h, 4738
  - uLongf
    - zconf.h, 4738
  - uncaughtException
    - decaf::lang::Thread::UncaughtExceptionHandler, 3906
  - UnknownHostException
    - decaf::net::UnknownHostException, 3907, 3908
  - UnknownServiceException
    - decaf::net::UnknownServiceException, 3910, 3911
  - unlock
    - activemq::core::MessageDispatchChannel, 2604
    - decaf::internal::util::concurrent::MutexImpl, 2789
    - decaf::internal::util::concurrent::SynchronizableImpl, 3713
    - decaf::io::InputStream, 2051
    - decaf::io::OutputStream, 2911
    - decaf::util::AbstractCollection, 190
    - decaf::util::concurrent::ConcurrentStlMap, 1246
    - decaf::util::concurrent::Lock, 2376
    - decaf::util::concurrent::locks::Lock, 2381
    - decaf::util::concurrent::locks::ReentrantLock, 3188
    - decaf::util::concurrent::Mutex, 2784
    - decaf::util::concurrent::Synchronizable, 3705
    - decaf::util::StlMap, 3612
    - decaf::util::StlQueue, 3620
  - unmarshal

- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 1525, 1526
- 3005
- URI
- activemq::wireformat::openwire::OpenWireFormat, 3923, 3924
- 2898
- URIEncoderDecoder
- activemq::wireformat::openwire::utils::BooleanStream, 3932
- 858
- activemq::wireformat::stomp::StompWireFormat, 3645
- URIHelper
- decaf::internal::net::URIHelper, 3936
- activemq::wireformat::WireFormat, 3978
- URIParam
- activemq::core::ActiveMQConstants, 309
- unmarshalList
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3005
- activemq::core::ActiveMQConstants::StaticInitializer, 3588
- unmarshalMap
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3006
- activemq::core::ActiveMQConstants::StaticInitializer, 3588
- unmarshalPrimitive
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3006
- activemq::transport::failover::URIPool, 3942
- unmarshalPrimitiveList
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3006
- URISyntaxException, 3948–3950
- unmarshalPrimitiveMap
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3007
- decaf::internal::net::URIType, 3954
- unpark
- URL
- decaf::util::concurrent::locks::LockSupport, 2385
- decaf::net::URL, 3961
- userID
- activemq::commands::Message, 2532
- unread
- decaf::io::PushbackInputStream, 3146
- userName
- unregisterFactory
- activemq::transport::TransportRegistry, 3904
- activemq::commands::ConnectionInfo, 1360
- activemq::wireformat::WireFormatRegistry, 4019
- ush
- zutil.h, 4745
- ushf
- zutil.h, 4745
- unsetenv
- decaf::lang::System, 3732
- UTFDataFormatException
- decaf::io::UTFDataFormatException, 3966, 3967
- UNSUBSCRIBE
- activemq::wireformat::stomp::StompCommandConstants, 3631
- UUID
- decaf::util::UUID, 3970
- unsubscribe
- activemq::cmsutil::PooledSession, 2966
- activemq::core::ActiveMQSession, 531
- cms::Session, 3376
- UnsupportedEncodingException
- decaf::io::UnsupportedEncodingException, 3913, 3914
- UnsupportedOperationException
- cms::UnsupportedOperationException, 3919
- decaf::lang::exceptions::UnsupportedOperationException, 3916, 3917
- update
- decaf::util::zip::Adler32, 723, 724
- decaf::util::zip::Checksum, 1143, 1144
- decaf::internal::net::URIHelper, 3939
- validate
- decaf::internal::net::URIEncoderDecoder, 3933
- validateAuthority
- decaf::internal::net::URIHelper, 3939
- validateFragment
- decaf::internal::net::URIHelper, 3939
- validatePath
- decaf::internal::net::URIHelper, 3939

- validateQuery
  - decaf::internal::net::URIHelper, 3940
- validateScheme
  - decaf::internal::net::URIHelper, 3940
- validateSimple
  - decaf::internal::net::URIEncoderDecoder, 3933
- validateSsp
  - decaf::internal::net::URIHelper, 3940
- validateUserinfo
  - decaf::internal::net::URIHelper, 3941
- value
  - activemq::commands::BrokerId, 871
  - activemq::commands::ConnectionId, 1330
  - activemq::commands::ConsumerId, 1434
  - activemq::commands::LocalTransactionId, 2350
  - activemq::commands::ProducerId, 3071
  - activemq::commands::SessionId, 3382
- valueOf
  - decaf::lang::Boolean, 854
  - decaf::lang::Byte, 966, 967
  - decaf::lang::Character, 1106
  - decaf::lang::Double, 1797
  - decaf::lang::Float, 1913
  - decaf::lang::Integer, 2089, 2090
  - decaf::lang::Long, 2432
  - decaf::lang::Short, 3447
  - decaf::util::concurrent::TimeUnit, 3815
- values
  - decaf::util::concurrent::ConcurrentStlMap, 1246
  - decaf::util::concurrent::TimeUnit, 3816
  - decaf::util::Map, 2470
  - decaf::util::StlMap, 3612
- variant
  - decaf::util::UUID, 3974
- verify
  - decaf::security::cert::Certificate, 1088
- version
  - decaf::util::UUID, 3974
- visit
  - activemq::commands::BrokerError, 866
  - activemq::commands::BrokerInfo, 902
  - activemq::commands::Command, 1198
  - activemq::commands::ConnectionControl, 1271
  - activemq::commands::ConnectionError, 1298
  - activemq::commands::ConnectionInfo, 1359
  - activemq::commands::ConsumerControl, 1404
  - activemq::commands::ConsumerInfo, 1465
  - activemq::commands::ControlCommand, 1495
  - activemq::commands::DestinationInfo, 1730
  - activemq::commands::FlushCommand, 1939
  - activemq::commands::KeepAliveInfo, 2268
  - activemq::commands::Message, 2530
  - activemq::commands::MessageAck, 2563
  - activemq::commands::MessageDispatch, 2597
  - activemq::commands::MessageDispatchNotification, 2633
  - activemq::commands::MessagePull, 2742
  - activemq::commands::ProducerAck, 3038
  - activemq::commands::ProducerInfo, 3099
  - activemq::commands::RemoveInfo, 3196
  - activemq::commands::RemoveSubscriptionInfo, 3225
  - activemq::commands::ReplayCommand, 3253
  - activemq::commands::Response, 3287
  - activemq::commands::SessionInfo, 3409
  - activemq::commands::ShutdownInfo, 3472
  - activemq::commands::TransactionInfo, 3850
  - activemq::commands::WireFormatInfo, 3991
- voidp
  - zconf.h, 4738
- voidpc
  - zconf.h, 4738
- voidpf
  - zconf.h, 4738
- w\_bits
  - internal\_state, 2120
- w\_mask
  - internal\_state, 2120
- w\_size
  - internal\_state, 2120
- wait
  - activemq::core::MessageDispatchChannel, 2604, 2605
  - decaf::internal::util::concurrent::ConditionImpl, 1258
  - decaf::internal::util::concurrent::SynchronizableImpl, 3713, 3714
  - decaf::io::InputStream, 2051, 2052
  - decaf::io::OutputStream, 2911, 2912
  - decaf::util::AbstractCollection, 190, 191
  - decaf::util::concurrent::ConcurrentStlMap, 1246, 1247
  - decaf::util::concurrent::Mutex, 2785, 2786

- decaf::util::concurrent::Synchronizable, 3706, 3708, 3709
- decaf::util::StlMap, 3612, 3613
- decaf::util::StlQueue, 3621, 3622
- WAIT\_INFINITE
  - Concurrent.h, 4893
- waitForSpace
  - activemq::util::MemoryUsage, 2514
  - activemq::util::Usage, 3965
- WAITING
  - decaf::lang::Thread, 3768
- wakeup
  - activemq::core::ActiveMQSession, 532
  - activemq::core::ActiveMQSessionExecutor, 536
  - activemq::threads::CompositeTaskRunner, 1225
  - activemq::threads::DedicatedTaskRunner, 1672
  - activemq::threads::TaskRunner, 3736
- want
  - gz\_state, 1977
- Warn
  - decaf::util::logging, 176
- warn
  - decaf::util::logging::SimpleLogger, 3502
- WARNING
  - decaf::util::logging::Level, 2336
- warning
  - decaf::util::logging::Logger, 2396
- was
  - inflate\_state, 2025
- wasPrepared
  - activemq::commands::JournalTransaction, 2241
- wbits
  - inflate\_state, 2025
- what
  - cms::CMSException, 1162
  - decaf::lang::Exception, 1837
- whave
  - inflate\_state, 2025
- WIN\_INIT
  - deflate.h, 4727
- window
  - inflate\_state, 2025
  - internal\_state, 2120
- window\_size
  - internal\_state, 2120
- windowSize
  - activemq::commands::ProducerInfo, 3100
- WireFormatInfo
  - activemq::commands::WireFormatInfo, 3984
- WireFormatInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::WireFormatInfo, 4009
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfo, 4001
  - activemq::wireformat::openwire::marshal::v3::WireFormatInfo, 4013
  - activemq::wireformat::openwire::marshal::v4::WireFormatInfo, 4005
  - activemq::wireformat::openwire::marshal::v5::WireFormatInfo, 3993
  - activemq::wireformat::openwire::marshal::v6::WireFormatInfo, 3997
- WireFormatNegotiator
  - activemq::wireformat::WireFormatNegotiator, 4016
- wnext
  - inflate\_state, 2025
- work
  - inflate\_state, 2025
- wrap
  - decaf::nio::ByteBuffer, 1054
  - decaf::nio::CharBuffer, 1133
  - decaf::nio::DoubleBuffer, 1818, 1819
  - decaf::nio::FloatBuffer, 1934
  - decaf::nio::IntBuffer, 2075
  - decaf::nio::LongBuffer, 2453
  - decaf::nio::ShortBuffer, 3468
  - inflate\_state, 2025
  - internal\_state, 2120
- write
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2869
  - decaf::internal::net::tcp::TcpSocket, 3746
  - decaf::internal::util::ByteArrayAdapter, 989
  - decaf::io::OutputStream, 2913, 2914
  - decaf::io::Writer, 4024–4026
- WRITE\_FAILURE
  - decaf::util::logging::ErrorManager, 1829
- writeBoolean
  - activemq::commands::ActiveMQBytesMessage, 244
  - activemq::commands::ActiveMQStreamMessage, 547
  - activemq::wireformat::openwire::utils::BooleanStream, 859
  - cms::BytesMessage, 1065
  - cms::StreamMessage, 3660
  - decaf::io::DataOutput, 1575
  - decaf::io::DataOutputStream, 1581
- writeByte
  - activemq::commands::ActiveMQBytesMessage, 244

- activemq::commands::ActiveMQStreamMessage, 547
- cms::BytesMessage, 1066
- cms::StreamMessage, 3660
- decaf::io::DataOutput, 1575
- decaf::io::DataOutputStream, 1582
- writeBytes
  - activemq::commands::ActiveMQBytesMessage, 245
  - activemq::commands::ActiveMQStreamMessage, 548
  - cms::BytesMessage, 1066
  - cms::StreamMessage, 3661
  - decaf::io::DataOutput, 1575
  - decaf::io::DataOutputStream, 1582
- writeChar
  - activemq::commands::ActiveMQBytesMessage, 245
  - activemq::commands::ActiveMQStreamMessage, 548
  - cms::BytesMessage, 1067
  - cms::StreamMessage, 3661
  - decaf::io::DataOutput, 1576
  - decaf::io::DataOutputStream, 1582
- writeChars
  - decaf::io::DataOutput, 1576
  - decaf::io::DataOutputStream, 1582
- WriteChecker
  - activemq::transport::inactivity::InactivityMonitor, 2007
  - activemq::transport::inactivity::WriteChecker, 4020
- writeDouble
  - activemq::commands::ActiveMQBytesMessage, 246
  - activemq::commands::ActiveMQStreamMessage, 549
  - cms::BytesMessage, 1067
  - cms::StreamMessage, 3662
  - decaf::io::DataOutput, 1576
  - decaf::io::DataOutputStream, 1582
- writeFloat
  - activemq::commands::ActiveMQBytesMessage, 246
  - activemq::commands::ActiveMQStreamMessage, 549
  - cms::BytesMessage, 1067
  - cms::StreamMessage, 3662
  - decaf::io::DataOutput, 1577
  - decaf::io::DataOutputStream, 1582
- writeInt
  - activemq::commands::ActiveMQBytesMessage, 246
- activemq::commands::ActiveMQStreamMessage, 549
- cms::BytesMessage, 1068
- cms::StreamMessage, 3662
- decaf::io::DataOutput, 1577
- decaf::io::DataOutputStream, 1582
- writeLock
  - decaf::util::concurrent::locks::ReadWriteLock, 3173
- writeLong
  - activemq::commands::ActiveMQBytesMessage, 247
  - activemq::commands::ActiveMQStreamMessage, 549
  - cms::BytesMessage, 1068
  - cms::StreamMessage, 3663
  - decaf::io::DataOutput, 1577
  - decaf::io::DataOutputStream, 1582
- Writer
  - decaf::io::Writer, 4022
- writeShort
  - activemq::commands::ActiveMQBytesMessage, 247
  - activemq::commands::ActiveMQStreamMessage, 550
  - cms::BytesMessage, 1068
  - cms::StreamMessage, 3663
  - decaf::io::DataOutput, 1577
  - decaf::io::DataOutputStream, 1582
- writeString
  - activemq::commands::ActiveMQBytesMessage, 247
  - activemq::commands::ActiveMQStreamMessage, 550
  - activemq::util::MarshallingSupport, 2495
  - cms::BytesMessage, 1069
  - cms::StreamMessage, 3663
- writeString16
  - activemq::util::MarshallingSupport, 2496
- writeString32
  - activemq::util::MarshallingSupport, 2496
- writeTo
  - decaf::io::ByteArrayOutputStream, 1030
- writeUnsignedShort
  - activemq::commands::ActiveMQBytesMessage, 247
  - activemq::commands::ActiveMQStreamMessage, 550
  - cms::BytesMessage, 1069
  - cms::StreamMessage, 3664
  - decaf::io::DataOutput, 1578
  - decaf::io::DataOutputStream, 1582
- writeUTF

activemq::commands::ActiveMQBytesMessage,	z_errmsg
248	zutil.h, 4745
cms::BytesMessage, 1069	Z_ERRNO
decaf::io::DataOutput, 1578	zlib.h, 4742
decaf::io::DataOutputStream, 1582	Z_FILTERED
written	zlib.h, 4742
decaf::io::DataOutputStream, 1582	Z_FINISH
wsz	zlib.h, 4742
inflate_state, 2025	Z_FIXED
	zlib.h, 4742
XATransactionId	Z_FULL_FLUSH
activemq::commands::XATransactionId,	zlib.h, 4742
4032	Z_HUFFMAN_ONLY
XATransactionIdMarshaller	zlib.h, 4742
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller,	Z_INTERNAL_ERROR
4049	zlib.h, 4742
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller,	Z_NO_FLUSH
4041	zlib.h, 4742
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller,	Z_NO_COMPRESSION
4053	zlib.h, 4742
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,	Z_NO_FLUSH
4045	zlib.h, 4742
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller,	Z_PARTIAL_FLUSH
4057	zlib.h, 4742
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,	Z_SYNC_FLUSH
4037	zconf.h, 4738
xflags	z_off_t
gz_header_s, 1975	zconf.h, 4738
XMLFormatter	Z_OK
decaf::util::logging::XMLFormatter, 4060	zlib.h, 4742
	Z_PARTIAL_FLUSH
yield	zlib.h, 4742
decaf::lang::Thread, 3773	Z_RLE
	zlib.h, 4742
Z_ASCII	z_stream
zlib.h, 4742	zlib.h, 4742
Z_BEST_COMPRESSION	Z_STREAM_END
zlib.h, 4742	zlib.h, 4742
Z_BEST_SPEED	Z_STREAM_ERROR
zlib.h, 4742	zlib.h, 4742
Z_BINARY	z_stream_s, 4062
zlib.h, 4742	adler, 4062
Z_BLOCK	avail_in, 4062
zlib.h, 4742	avail_out, 4062
Z_BUF_ERROR	data_type, 4062
zlib.h, 4742	msg, 4062
Z_DATA_ERROR	next_in, 4062
zlib.h, 4742	next_out, 4062
Z_DEFAULT_COMPRESSION	opaque, 4062
zlib.h, 4742	reserved, 4062
Z_DEFAULT_STRATEGY	state, 4062
zlib.h, 4742	total_in, 4062
Z_DEFLATED	total_out, 4062
zlib.h, 4742	zalloc, 4062

- zfree, 4062
- z\_streamp
  - zlib.h, 4742
- Z\_SYNC\_FLUSH
  - zlib.h, 4742
- Z\_TEXT
  - zlib.h, 4742
- Z\_TREES
  - zlib.h, 4742
- Z\_UNKNOWN
  - zlib.h, 4742
- Z\_VERSION\_ERROR
  - zlib.h, 4742
- ZALLOC
  - zutil.h, 4745
- zalloc
  - z\_stream\_s, 4062
- zconf.h
  - Byte, 4738
  - Bytef, 4738
  - charf, 4738
  - const, 4738
  - intf, 4738
  - MAX\_MEM\_LEVEL, 4738
  - MAX\_WBITS, 4738
  - OF, 4738
  - SEEK\_CUR, 4738
  - SEEK\_END, 4738
  - SEEK\_SET, 4738
  - uInt, 4738
  - uIntf, 4738
  - uLong, 4738
  - uLongf, 4738
  - voidp, 4738
  - voidpc, 4738
  - voidpf, 4738
  - z\_off64\_t, 4738
  - z\_off\_t, 4738
  - ZEXTERN, 4738
- ZEXTERN
  - zconf.h, 4738
- ZFREE
  - zutil.h, 4745
- zfree
  - z\_stream\_s, 4062
- ZipException
  - decaf::util::zip::ZipException, 4064, 4065
- zlib.h
  - deflateInit, 4741
  - deflateInit2, 4741
  - gz\_header, 4742
  - gz\_headerp, 4742
  - gzFile, 4742
  - inflateBackInit, 4741
  - inflateInit, 4741
  - inflateInit2, 4742
  - OF, 4742
  - Z\_ASCII, 4742
  - Z\_BEST\_COMPRESSION, 4742
  - Z\_BEST\_SPEED, 4742
  - Z\_BINARY, 4742
  - Z\_BLOCK, 4742
  - Z\_BUF\_ERROR, 4742
  - Z\_DATA\_ERROR, 4742
  - Z\_DEFAULT\_COMPRESSION, 4742
  - Z\_DEFAULT\_STRATEGY, 4742
  - Z\_DEFLATED, 4742
  - Z\_ERRNO, 4742
  - Z\_FILTERED, 4742
  - Z\_FINISH, 4742
  - Z\_FIXED, 4742
  - Z\_FULL\_FLUSH, 4742
  - Z\_HUFFMAN\_ONLY, 4742
  - Z\_MEM\_ERROR, 4742
  - Z\_NEED\_DICT, 4742
  - Z\_NO\_COMPRESSION, 4742
  - Z\_NO\_FLUSH, 4742
  - Z\_NULL, 4742
  - Z\_OK, 4742
  - Z\_PARTIAL\_FLUSH, 4742
  - Z\_RLE, 4742
  - z\_stream, 4742
  - Z\_STREAM\_END, 4742
  - Z\_STREAM\_ERROR, 4742
  - z\_streamp, 4742
  - Z\_SYNC\_FLUSH, 4742
  - Z\_TEXT, 4742
  - Z\_TREES, 4742
  - Z\_UNKNOWN, 4742
  - Z\_VERSION\_ERROR, 4742
  - ZLIB\_VER\_MAJOR, 4742
  - ZLIB\_VER\_MINOR, 4742
  - ZLIB\_VER\_REVISION, 4742
  - ZLIB\_VER\_SUBREVISION, 4742
  - ZLIB\_VERNUM, 4742
  - ZLIB\_VERSION, 4742
  - zlib\_version, 4742
- ZLIB\_INTERNAL
  - gzguts.h, 4729
  - zutil.h, 4745
- ZLIB\_VER\_MAJOR
  - zlib.h, 4742
- ZLIB\_VER\_MINOR
  - zlib.h, 4742
- ZLIB\_VER\_REVISION
  - zlib.h, 4742
- ZLIB\_VER\_SUBREVISION
  - zlib.h, 4742

---

ZLIB\_VERNUM  
  zlib.h, 4742  
ZLIB\_VERSION  
  zlib.h, 4742  
zlib\_version  
  zlib.h, 4742  
zstrerror  
  gzguts.h, 4729  
zutil.h  
  Assert, 4745  
  DEF\_MEM\_LEVEL, 4745  
  DEF\_WBITS, 4745  
  DYN\_TREES, 4745  
  ERR\_MSG, 4745  
  ERR\_RETURN, 4745  
  F\_OPEN, 4745  
  MAX\_MATCH, 4745  
  MIN\_MATCH, 4745  
  OF, 4745  
  OS\_CODE, 4745  
  PRESET\_DICT, 4745  
  STATIC\_TREES, 4745  
  STORED\_BLOCK, 4745  
  Trace, 4745  
  Tracec, 4745  
  Tracecv, 4745  
  Tracev, 4745  
  Tracevv, 4745  
  TRY\_FREE, 4745  
  uch, 4745  
  uchf, 4745  
  ulg, 4745  
  ush, 4745  
  ushf, 4745  
  z\_errmsg, 4745  
  ZALLOC, 4745  
  ZFREE, 4745  
  ZLIB\_INTERNAL, 4745